Design and implementation of an intelligent DNS management system

Chien-Liang Liu\textsuperscript{a}, Shian-Shyong Tseng\textsuperscript{a,}\textsuperscript{*}, Chang-Sheng Chen\textsuperscript{b}

\textsuperscript{a}Department of Computer and Information Science, National Chiao-Tung University, 1001, Ta Hsueh Road, HsinChu Taiwan 300
\textsuperscript{b}Computer and Network Center, National Chiao-Tung University, 1001, Ta Hsueh Road, HsinChu Taiwan 300

Abstract

In this paper, we focus on the design and building of iDNS-MS by using knowledge-based system and ontological engineering technologies. iDNS-MS, developed by using web interface and expert system technology, is a unifying environment for providing plausible answers to help solve the complex DNS management problems or alleviate these DNS administration loadings. In iDNS-MS, we propose an ontology-driven model to elicit rules from a previously built DNS ontology and construct an object-oriented knowledge base. The whole process of the model consists of ontology construction, knowledge class organization and facts/rules loading phases. Ontology construction phase is used to construct the domain ontology, knowledge class organization phase is used to organize the relationships between the knowledge classes, and facts/rules loading phase is used to fill in the facts/rules of knowledge classes extracted from domain experts. In addition, we adopt DRAMA/NORM development environment as the expert system shell to design and implement a unifying framework (e.g. DNS-related problem diagnosis, planning, tutoring, etc.) for supporting intelligent DNS management. According to our experimental results, the paradigm of using DNS ontology to build iDNS-MS works good and effective. iDNS-MS will benefit the sharing and reusing of global DNS knowledge, the reduction of people’s time to learn DNS management, the ease of DNS configuration and planning, and the improvement on DNS and network operation.

\textsuperscript{2}2004 Elsevier Ltd. All rights reserved.

Keywords: DNS ontology; Ontology-driven; Knowledge-based system; iDNS-MS

1. Introduction

As with the popularity of Internet, the expert system (ES) technology has been applied to various applications in internetworking services, producing a considerable amount of knowledge as a by-product. Such knowledge compiled through internetworking applications can offer learning opportunities to the Internet communities for knowledge sharing and improving the management of the Internet. The main purpose of this paper is to describe the development process of an intelligent DNS (Albitz & Liu, 2001) Management System, called iDNS-MS.

The development of iDNS-MS is marked by many characteristics. First, the DNS is an essential part of the Internet infrastructure. All internetworking traffic might seem to fail if the related DNS servers do not work properly as expected. Second, lots of novice DNS administrators (e.g. SOHO people, etc.) do not know the theoretical and practical knowledge of DNS system very well. Thirdly, many administrators that manage a small scale of network lack the experiences for dealing with the global distributed Internet traffic. Some serious problems (e.g. using buggy versions of DNS software, inappropriate configuration or planning problems, etc.) have not been identified or even been ignored on these sites. Finally, given the importance of DNS servers, direct or indirect attacks on the DNS systems are common (Brownlee, Claffy, & Nemeth, 2001; Koh, 2001).

In this paper, we focus on the design and building of iDNS-MS (Chen, Tseng, & Liu, 2003), by using knowledge-based system (KBS) and ontological engineering technologies. iDNS-MS, developed by using web interface and expert system technology, is a unifying environment for providing plausible answers to help solve the complex DNS management problems or alleviate these DNS administration loadings.

In iDNS-MS, we adopt DRAMA/NORM (Lin, Tseng, & Tsai, 2003) as the expert system shell because of its client–server architecture and the object-oriented knowledge base structure. The client–server feature of DRAMA/NORM makes it easy to develop KBS for supporting intelligent DNS management through web interface. On the other hand, the knowledge model of DRAMA/NORM is based on
knowledge classes (KCs), which are like the concepts of ontology. Therefore, the transformation between the ontology concepts and the KCs is easy. In NORM, a KC represents a kind of concept that people realize. It consists of rules, facts declarations and relations (with other KCs). The facts and rules denote the internal characteristics of the KC and the relations between the KCs simulate the interaction of the concepts. In addition, because of the object-oriented knowledge base structure, the knowledge can be modularly managed. There are many advantages of using such a modular knowledge base design. First, the knowledge base is partitioned into general clusters of concepts and rules are grouped into sets of specific concept domains. Thus, it provides a logical partitioning of the Rule-Base (RB), which facilitates the management of rules in each KC. Second, it is easy to reuse existing rules based on modular knowledge base design. Therefore, this can help provide personalized service for different users.

In this paper, we propose an ontology-driven model to help extract KBS rules from DNS problem cases. There are three phases in the ontology-driven model: ontology construction phase, KC organization phase and facts/rules loading phase. Ontology construction phase is used to construct the domain ontology, KC organization phase is used to organize the relationship between the KCs, and facts/rules loading phase is used to fill in the facts/rules of KCs extracted from domain experts. As mentioned in Chandrasekaran, Josephson, and Benjamins (1999), the role of ontologies is to capture domain knowledge and provide a commonly agreed upon understanding of a domain; however, like many real-world applications, most problems in DNS domain could be easily addressed by using rules. However, rules extraction from domain experts is not necessarily a straightforward job; we often need some knowledge acquisition processes to help achieve the goal. The main functionality of ontology-driven model is to help the KEs to extract the rules with the help of ontology. In essence, ontology representation is suitable for communications and natural for human thinking, meanwhile rule representation is powerful for machine to manipulate the concepts. Ontology-driven model could facilitate the transformation of ontology representation and rule representation.

The iDNS-MS system began to offer public access service from the early days of May, 2003. By statistical analysis, we have over 10,000 access counts from over 3000 different IP addresses in a nearly 3 month interval (i.e. from July 7 to September 30, 2003). According to the experimental results, the paradigm of using DNS ontology to build iDNS-MS works good and effective. iDNS-MS will benefit the sharing and reusing of global DNS knowledge, the reduction of people’s time to learn DNS management, the ease of DNS configuration and planning, and the improvement on DNS and network operation. It is supposed that, the same approach could be adaptively modified to other problem domains for knowledge base construction.

2. Background and related work

In the following, we will briefly mention the preliminaries on the related work.

2.1. DNS as a building block of the internetworking environment

The DNS (Albitz & Liu, 2001) is responsible for translating between hostnames used by people and corresponding IP addresses needed by software. The mapping of data is stored in a tree-structured distributed database where each nameserver is authoritative (responsible) for a portion of the naming hierarchy tree.

In essence, the distributed mechanism of DNS is a double-edged sword; it allows DNS to scale to Internet size, but it also allows for incredible mis-configurations and more difficulty on debugging (and tracing). DNS problem domain is very complex and varies greatly on different sites because too many things, like environments (e.g. IPv4/IPv6, multilingual, etc.), management strategies and resources, need to be considered. Moreover, as shown in Fig. 1, more and more organizations could implement DNS-based security systems (e.g. firewalls, anti-virus systems, anti-spam mail systems, anti-erotic proxies, etc.) for supporting functionalities such as content filtering, mail routing, network load sharing, etc. These implicit DNS-based applications might have some direct interactions or indirect side effects with the related DNS systems and the overall Internet. Therefore, these make the DNS management jobs (and the related network management) more complicated and become even more important than ever before.

2.2. Problem domain

Even though DNS is so important to network operation, few DNS administrators have the expertise to do the jobs well. Improperly configured (or planned) DNS traffic or intent anomalous activities (i.e. DNS domain zone scanning, SPAM mails, intrusion attempts, etc.) from local and remote sites might greatly affect the operation of local DNS systems and the related network from time to time. The phenomenon is especially obvious between novice 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 (MenandMice, 2002) showed that nearly 70% of the DNS servers of commercial sites (e.g. ‘.COM’ Zones) have some configuration errors.

On the other hand, in Brownlee et al. (2001), the authors passively measured the performance of one of the 13 root servers: f.root-servers.net. These measurements showed an astounding number of bogus queries: from 60 to 85% of observed queries were repeated from the same host within the measurement interval. Over 14% of a root server’s query load is due to queries that violate the DNS specification.
Denial-of-service (DoS) attacks, using root servers as reflectors toward a victim network, are common. From time to time, we could easily find that there is a lot of anomalous DNS traffic by mal-formed DNS software programs or inappropriately configured DNS server hosts on many parts of Internet.

2.3. Overview of the expert system development environment—DRAMA

In this paper, we use DRAMA (Lin et al., 2003) as an expert system shell because of its client–server architecture and the object-oriented knowledge base structure. As shown in Fig. 2, the purpose of the DRAMA’s server is to load, manage and use the knowledge bases according to the knowledge service that users need. DRAMA’s server contains many different RBs, and provides different APIs for the AP servers to connect. The AP server employs DRAMA’s APIs to provide user-friendly web pages for users to use expert systems. Based on the client–server architecture, it thus becomes very easy for us to develop KBS for supporting intelligent DNS management through WWW interface.

3. DNS ontology and DRAMA model

For KBSs, in particular, the high cost of knowledge acquisition makes reuse essential (Lopez, Gomez-Perez, Sierra, & Sierra, 1999). It is well known that the knowledge acquisition (KA) is usually the bottleneck of building a KBS. The process of the KA is to transfer domain knowledge into knowledge bases. In general, knowledge acquisition 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 this paper, we propose an ontology-driven model to facilitate rules extraction. In the following, we will describe the rationale behind.

3.1. Use case modeling and DNS ontology building

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 (Chandrasekaran et al., 1999). The role of ontologies is to capture domain knowledge and provide a commonly agree 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 (Heijst, Schreiber, & Wielinga, 1997). 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. Therefore, 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).

As shown in Fig. 3 (Chen et al., 2003), we extract the concepts and attributes by using a hybrid method consisting
of the brainstorming and use case modeling (Cockburn, 1997). The power of a few critical cases described in terms of relevant attributes to build domain ontologies is remarkable. This is because it is often easier and more accurate for the experts to provide critical cases and it would 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 ontological components easily. Hence, use cases analysis is adequate for our DNS knowledge acquisition.

3.2. DNS domain knowledge and ontology

Just like the concept of object-oriented programming, we could view all the entities in the real world as concepts and it is natural for us to model the world using concepts hierarchy. For example, a DNS server is a concept, and it contains attributes or slots: hostld (i.e. IPv4/IPv6 address), serverType (e.g. authoritative server, caching server, etc.), hostInventory (e.g. 1 Gb RAM, 2.80 GHz CPU, 100 Mb Ethernet, etc.), dnsServerSoftware (e.g. FreeBSD 4.9, BIND-9.2.3, etc.), etc. Furthermore, people tend to group the knowledge and build structural information when they learn new concepts. The grouped knowledge could be viewed as a bigger concept as well. For example, both Single-Point-Of-Failure (SPOF) and DNS configuration error (e.g. lamed DNS servers) are typical types of the DNS availability problems. Hence, the SPOF concept (and lamed-server concept, too) inherits the DNS availability concept, and there exists an Is_a relationship between them. Similarly, when we learn DNS-related issues, the same approach could be applied to cover other issues including DNS securities, performance, etc. On the other hand, people often need to reference other concepts when learning specific concepts. For example, when we refer to the DHCP-DNS attack concept, we will also reference the concept about dynamic host configuration (i.e. DNS dynamic update) via the DHCP mechanism. By combining these, we could group all DNS-related knowledge together and build a concept hierarchy about DNS.

In essence, ontology representation is suitable for communications and natural for human thinking, meanwhile rule representation is powerful for machine to manipulate the concepts. As described above, ontology could be used to model the concept hierarchy and relationships between concepts. However, it is not easy to model the behavior of concepts using ontology only. When the problem domain can be described clearly and well modeled, it is much easier to build a RB expert system because many tools (called expert system shells) can offer assistances. Hence, in practice, rule-based representation is more suitable for building applications. On the other hand, since most applications need complex rules to solve real world problems, the information captured in an ontology for the problem domain could become very helpful for rule extractions when building complex systems.

For many people (e.g. DNS beginners, etc.), information of DNS taxonomy will help them understand operating details of the DNS and describe encountered problems more explicitly. Fig. 4 shows a snapshot of DNS ontology (Chen et al., 2003). Three types of relationships and one constraint are described as follows

- **Three types of relationships**: (1) *Is_a* is a generalization relationship, which could be used to describe the concept taxonomies in the class hierarchy. For example, either a master (class) or a slave DNS server (class) is a kind of authoritative DNS server (class). (2) *Rel* (i.e. related-to relationship) denotes that there exists some relationship between these terms. For example, we could use *Rel* relationship to denote that the DNS security class is related to the DNS server class. (3) *Case* is ‘case of’ relationship. For example, SPOF concept is one of the cases leading to ‘DNS availability’ concept.

- **Identification of constraints**: (1) pre-requisite constraint: one term/relationship depends upon another. For example, the SPOF concept depends on many concepts including: ‘Single Network’, ‘Single Router’ and ‘Single Server’.

3.3. New object-oriented rule model and knowledge class

Even though rule-based systems are powerful enough in many applications, they usually have several disadvantages such as the weak ability of incremental construction of knowledge (Lee and O’Keefe, 1996) in maintenance and construction. NORM, combining the advantages of ontology and rule-based system, could be used to wrap and manipulate the concepts using rules.

The development of NORM (Lin et al., 2003) is primarily based on how human learn new concepts. Fig. 5 shows the
model of NORM that contains KCs and the relationships among them. In NORM, a KC represents a kind of concept that people realize. It consists of rules, relations (with other KCs) and fact declarations. The function of relationships is for facilitating the mapping between different KCs whenever necessary. For example, if we want to learn another new concept, we may reference certain concepts that we have in our mind. By this way, we can build the new concept quickly. Thus, a RB can record various knowledge concepts in a specific domain and each KC in the RB represents different concept of the domain knowledge.

The working model of Fig. 6 is composed of different KCs and the transferences (e.g. Trigger, Require, Reference, and Extension-of, etc.) between KCs. The relationships between KCs are divided into two kinds—dynamic and static. The former two (e.g. Trigger and Acquire) are dynamic because they are activated conditionally in the action part of a rule, while the latter two are static.

3.4. Ontology-driven model for rule extraction

For dealing with maintenance issues, KCs could group the related knowledge together to improve the maintenance of the rules. As for construction issues, ontology could still play an important role even though it is not easy to extract rules directly from the ontology. First, as described above, the ontology could be used as the common language between knowledge engineers and domain experts. Second, the ontology provides the hints of rules extraction to assist knowledge engineers in interviewing domain experts.
As shown in Fig. 7, we propose an ontology-driven model for rules extraction. The whole process is described as follows.

### 3.4.1. Ontology construction phase

The first phase is ontology construction. Up till now, the ontology building process is still a craft rather than an engineering activity (Heijst et al., 1997). Each development team usually follows its own set of principles, design criteria and phases on the ontology development process. In Chen et al. (2003), we proposed to construct ontology by using a hybrid method consisting of the brainstorming and use case modeling (Cockburn, 1997). Fig. 4 shows a snapshot of the DNS ontology. The DNS construction algorithm is summarized as follows:

**DNS ontology constructing algorithm**

**Input:** every kind of DNS cases.

**Output:** DNS Ontology.

**Step 1:** build the Skeleton DNS ontology (top-down)

**Step 2:** initiate (or conduct) use case modeling

**Step 3:** conduct the attributes and relation extraction.

**Step 4:** merge the ontological components collected in Step 1 and Step 3 above.

**Step 5:** experts verify the ontology.

**Step 6:** after experts’ verification, the DNS ontology is constructed to cover DNS domain knowledge.

### 3.4.2. Knowledge class organization phase

As described above, since the KC of NORM knowledge model is based on the concepts, the transformation between the ontology concept class and the KC could be very straightforward. However, generally speaking, the knowledge for specific domain is usually large and we need some directions to narrow down the scope. In other words, the major problem on ‘which concept classes need to be transferred’ should be determined. The ontology relationships could give us some hints during the transformation. For example, the DNS diagnosis application focuses on the DNS problems, so the knowledge engineer needs to explore the DNS related problems first. Therefore, we could transfer the major ontology concept classes about DNS diagnosis into the corresponding KCs as described in Fig. 8.

In the process of DNS construction, we should consider DNS issues including availability, performance and registration, etc. Fig. 8 shows the inference scheme of diagnostic examples about DNS-related mailing problems. The rectangles mean KC’s in NORM and the rounded rectangles mean cases of some particular KC’s. In addition, the solid lines indicate relations of the KC’s and their correlated cases.

As specified in Fig. 4, the ‘Rel’ relationships in DNS ontology show the DNS-related issues during building a DNS server. We may need to decompose the concepts into smaller sub-concepts to help analyze the cases. In this paper, a top-down approach is adopted to explore the knowledge; that is, we start from general concepts and then drill down to specific concepts. In addition, the relationships between KCs are constructed as well. For example, there exists an ‘is_a’ relationship between the DNS availability concept and the SPOF concept. Therefore, when considering the DNS availability issue, we should take measures to avoid the SPOF problem. The whole process could be summarized as follows.

**Ontology to KC transformation algorithm**

**Input:** DNS ontology

**Output:** DNS KCs and the relationships of KCs

**Step 1:** transfer the needed ontology concepts into KCs:

for each DNS ontology concept, we could transfer the concept into the KC.

![Fig. 5. New Object-oriented rule base model.](image)

![Fig. 6. The knowledge class in a rule base.](image)
Step 2: define or identify the relationships between the KCs.

Step 2.1: if there is an Is_a relationship between concept Ontology_X and concept Ontology_Y, we could infer that concept Ontology_X inherits concept Ontology_Y and that introduces the Extension-of relationship between the KCs, KC_X and KC_Y.

Step 2.2: if there is a Rel relationship between concepts Ontology_X and Ontology_Y, we could infer that when we talk about Ontology_X, we may talk about Ontology_Y as well. Therefore, that introduces the Acquire relationship between the KCs KC_X and KC_Y.

Step 2.3: if there is a Rel relationship between concept Ontology_X and concept Ontology_Y, and Case relationship between concept Ontology_Y and concept Ontology_Z, then that means concept Ontology_X may reference Ontology_Z. So, that introduces the Reference relationship the KCs KC_X and KC_Z.

Step 2.4: if there exists other relationship between any pair of concept Ontology_X and concept Ontology_Y, KEs should contact the domain experts for further analyzing.

3.4.3. Facts/rules loading phase
As described above, the KC consists of rules, relations (with other KCs) and fact declarations. After the KC
organization stage, the KCs hierarchy is built but the rules and facts of the KCs are still empty. Next, in the facts/rules-loading phase, we will load the facts and rules into the corresponding KCs. In this phase, we could further divide the stages into two sub-phases.

**Cases → Attribute ordering table**

As mentioned in (Gaines & Shaw, 1992), 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. In this paper, we make use of repertory grid like concept to help elicit knowledge. Table 1 shows the four cases resulting in SPOF. Knowledge Engineers construct the empty attribute ordering table first and then interview the domain experts to fill in the table with appropriate value. The value indicates whether the case relates to the attributes or not. Table 2 shows the ordering table of single server and single network.

**Attribute ordering table → Pseudo rules**

After the generation of repertory grid, we need to analyze the higher relative attributes of the cases. For example, when we refer to NS record attribute, we will refer the number of NS record and the IP address of each NS record as well. That is, we would like to find out the attribute/value pair of the facts. As described above, ontology contains the attributes of the concepts. Therefore, KEs could conduct the ontology to construct the empty attribute table for the higher relative slot of repertory grid and then interview the domain experts to fill in the values of the attributes. Tables 3 and 4 show the attribute/value pair tables for single network and single server, respectively. Finally, KEs could generate the pseudo rules, as shown in Table 5, based on the attribute/value pair.

In practice, while the KEs often do not have much knowledge about the problem domain, the domain experts usually do not have the programming concepts. Pseudo rules, viewed as the bridge between the domain experts and the KEs, are abstractions of the cases. They are understandable for the KEs and easier to be verified by the domain experts. If there is anything wrong, the domain experts could tell the KEs to modify the pseudo rules.

**Knowledge class facts/rules loading algorithm**

*Input*: DNS ontology

*Output*: DNS Knowledge Class with facts and rules

*Step 1*: find out the ontology concepts that contain ‘Case’ relationship.

*Step 2*: choose exemplary attributes that could characterize the domain.

*Step 3*: interview domain experts to rate each case based on the attributes. The value of the slot ranges from 1 to 5, where 5 means highly related with the construct while 1 means lowly related.

*Step 4*: find the highly related constructs and further analyze.

---

### Table 1

**SPOF case description**

<table>
<thead>
<tr>
<th>Description</th>
<th>DNS server is the infrastructure of the Internet, and if your DNS is unavailable at all times, the services depending on DNS (such as WWW, Email, etc.) will fail as well.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Case no.</td>
<td>Case name</td>
</tr>
<tr>
<td>-------------</td>
<td>-----------</td>
</tr>
<tr>
<td>Case 1</td>
<td>Single DNS Server</td>
</tr>
<tr>
<td>Case 2</td>
<td>Improper DNS configuration</td>
</tr>
<tr>
<td>Case 3</td>
<td>The same physical position</td>
</tr>
<tr>
<td>Case 4</td>
<td>The same router</td>
</tr>
</tbody>
</table>

### Table 2

**Attribute ordering table for single server and single server cases**

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Single server</th>
<th>Single network</th>
</tr>
</thead>
<tbody>
<tr>
<td>NS record</td>
<td>5</td>
<td>1</td>
</tr>
<tr>
<td>MX record</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>A record</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>PTR record</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>SOA record</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Physical location</td>
<td>1</td>
<td>5</td>
</tr>
<tr>
<td>CNAME</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Zone data</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

### Table 3

**Attributes and values of NS records for single server**

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>The Number of NS record</td>
<td>&lt; 2</td>
</tr>
<tr>
<td>The IP address of NS records</td>
<td>Master DNS and Slave DNS are not alive.</td>
</tr>
</tbody>
</table>
Step 4.1: conduct the ontology to construct the attribute tables.

Step 4.2: interview the domain experts to fill in the values of the attribute tables.

Step 5: generate pseudo rules, where facts coming from the attributes/values pairs of step 4.1.

Step 6: verify the pseudo rules by domain experts and ask the KEs to modify the pseudo rules if needed.

4. System architecture of iDNS-MS

DNS is one of the key components of the Internet infrastructure. Many Internet services (e.g., WWW, Email, etc.) rely on the proper operation of DNS. If DNS fails, these services might suffer from being unable to operate smoothly as well. Although DNS is so important to network operation today, many novice DNS administrators often do not know whether their DNS servers work well. It is expected that iDNS-MS could at least achieve two goals. First, for those who want to build up new DNS servers, iDNS-MS could provide DNS-related knowledge for them. Second, for those who want to check whether their DNS works well or not, iDNS-MS could help diagnose their DNS servers.

### 4.1. Overview of iDNS-MS system

As shown in Fig. 9, our system contains four subsystems: DNS Tutoring, DNS Diagnosis, DNS Planning, and DNS Configuration. The Tutoring subsystem focuses on basic knowledge and FAQs (Frequently Asked Questions) of DNS. It is expected that the Tutoring subsystem could help users have correct DNS knowledge and configure their DNS hosts well. The Diagnosis subsystem is used to diagnose existing DNS hosts by querying the DNS servers directly or by getting the related DNS information through a set of questions indirectly. If there is anything wrong, the system could remind the user and provide suggestions for fixing the problems. The Planning subsystem is mainly used for those wishing to reconstruct their DNS server(s) due to system environment changes (e.g., the number of the user increases, the existing DNS needs security enhancement, etc.). The Configuration subsystem is used to parse the users’ configuration files and report the configuration error(s). In the following, without loss of generality, we would focus on the DNS Diagnosis subsystem, which makes extensive use of the expert system approach.

### 4.2. System architecture of iDNS-MS

Fig. 10 shows the system architecture of iDNS-MS. As described in Section 3.4, the ontology-driven model is used to help extract the knowledge from domain experts and then store the knowledge into the KBS (DRAMA/NORM). Because of the object-oriented structure of DRAMA/NORM, the knowledge can be modularly managed. In the system design, we conducted some experts to understand users’ requirements, including system user interface, DNS frequently asked questions, and what resources they could make available, etc. In addition, to meet the requirement of the DNS administrators, the system design is flexible. For example, for many users, they do not know how to describe their DNS problems, and they would like to know whether their DNSs work well or not. What they could provide is the DNS domain names, and we have to collect other related information by querying the DNS server directly. But for some experienced DNS administrators, the questions and answer model could let us get more detailed information. The more detailed information the system

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Master DNS server location and Slave DNS server location</td>
<td>In the same network location</td>
</tr>
<tr>
<td>Master DNS server location and Slave DNS server location</td>
<td>Behind the same router</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Case name</th>
<th>Rule</th>
</tr>
</thead>
<tbody>
<tr>
<td>Single DNS server</td>
<td>If number of NS Record &lt; 2, Then SPOF</td>
</tr>
<tr>
<td>Improper DNS configuration</td>
<td>If Master DNS and Slave DNS are not live, Then SPOF</td>
</tr>
<tr>
<td>The same physical position</td>
<td>If master DNS Server and slave DNS server are in the same network location, Then SPOF</td>
</tr>
<tr>
<td>The same router</td>
<td>If the location of Master and Slave DNS servers are behind the same router, Then SPOF</td>
</tr>
</tbody>
</table>
collects, the more accurate suggestions could be provided by our system. For diagnosis system, two mechanisms are used to collect the facts of the rules. The first one queries the DNS server directly, while the second one gets the information through questions. These collected facts will be sent to the inference engine and then the inference results will return to the web interface.

4.3. Knowledge base system

In traditional RB expert system, the RB consists of all rules and facts. The system needs to go through every matching rule when the inference engine is working. This might become inefficient when the number of rules and facts become large. Therefore, many researches aim to improve the maintenance of rule-based expert system by incorporating the objected-oriented approach. DRAMA/NORM adopts KC to manipulate the knowledge and loads only the required KCs. That could simplify the rules management and improve the efficiency of the KBS. In essence, each knowledge module is corresponding to the KC structure of DRAMA. There are many advantages of using such a modular knowledge base design. First, the knowledge base is partitioned into general clusters of concepts and rules are grouped into sets of specific concept domains. Thus, it provides a logical partitioning of the RB, which facilitates the management of rules in each KC. Second, it is easy to reuse existing rules based on modular knowledge base design. Therefore, we can provide personalized service for different users.

In addition, the design of KCs takes into account knowledge reuse. For example, the rule,

If $TTL_1 \neq TTL_2$ then $LameServer = true$,

is located in ‘DNS Registration’ KC and it needs the facts of DNS server KC. In principle, the facts $TTL_1$ and $TTL_2$ in the ‘DNS Server’ KC will be taken (transferred) to the ‘DNS Registration’ KC. Therefore, as shown in Fig. 11, there is a relation Acquire between them.

- Rule : If $TTL_1 \neq TTL_2$ then $LameServer = true$

4.4. Web interface

In our system, two mechanisms are used to collect the user’s DNS server information

- If the user knows only the domain name, we will perform query operation to collect the DNS server information.
- If the user could provide more detailed information about the DNS environment, the questions and answer model is used to help acquire the user’s DNS information.

In addition, we adopt Model-View-Controller design pattern (Gamma, Helm, Johnson, & Vlissides, 1995) to separate core business model functionality from the presentation and control logic. Such separation allows multiple views to share the same enterprise data model, which makes it easier to implement, test, and maintain. The view section, made up by JSP files, is used to collect users’ DNS server information and display the diagnosis results back to the users. The collected information, gathering directly by querying or indirectly by asking questions, will be stored in the model section, the JavaBean, which is translated from the DNS ontology. The controller is...
composed by Java servlets. Based on the user interactions and the outcome of the inference engine, the controller responds by selecting an appropriate view.

5. Implementation and evaluation

5.1. System implementation

For building a web-based expert system, we use DRAMA as the expert system shell because of its client–server architecture and the object-oriented knowledge base structure. DRAMA is implemented by Java language and it uses Java RMI technique; thus, a web server can be a client of DRAMA by calling remote functions in DRAMA server. At the time of the writing: (1) the main operating system deployed is Linux Redhat 9.0; (2) the expert system tool is DRAMA 2.0; (3) the web server packages deployed are Apache 1.3.26, Tomcat 4.1.12. Interested users could refer to the web site (http://idns-kde.nctu.edu.tw) for further details.

As mentioned above, our iDNS-MS consists of many types of subsystems, namely, diagnosis, planning, tutoring, and configuration, etc. Users can connect the site to get advices on DNS management. However, for simplicity and ease of illustration, we will only focus on the diagnostic subsystem in the following. For example, as shown in Fig. 12, there are three diagnosis facilities for users to choose

- DNS on-line test: this will test the DNS servers that are supposed to be responsible for the domain zone. All users need to do is to enter a domain name and to select the DNS server. Then our system will conduct the required DNS queries and collect the information about this server from Internet automatically. After that, it will send the information to the server of DRAMA for inference. Finally, the server of DRAMA will return the inference results to the users via the web server.
- DNS off-line debugging: this facility is designed for the users, especially for DNS beginners, who want to build DNS servers but cannot make the DNS work by themselves. When the users have set the system files, they can upload these files to the system for verifying and debugging. Our system will point out the errors with colorful words and provide the possible way(s) to correct.
- Diagnosis of DNS-related mail problem: this subsystem will provide diagnosis services for people with the mail delivery problems related to DNS. Since there are many possible situations, we need to communicate with each user interactively with a list of questions to help identify and collect the facts that are needed for putting into the knowledge base and for later inference. After that, the system could provide plausible answers for the users to fix the problems on the related mail servers and/or DNS servers.

Fig. 12. The DNS diagnostic subsystem.
5.2. Typical diagnostic examples

Among the DNS-related problems, mail delivery problems are the most concerned. When users encounter mail delivery problems (that might involve DNS) and have no ideas what is really going on, they can use the diagnostic subsystem of iDNS-MS for getting plausible solutions. As could be derived from Fig. 8, users will be asked about which diagnosis type to try in the first place. If it is about DNS-related mailing problems, the Mail Delivery KC is triggered. Next, according to the cases, our system will further try identifying the problem(s) by asking the particular users with a list of questions about the status of related mail server and the corresponding DNS server(s).

As shown in Fig. 8, for identifying possible ‘No-existent reverse DNS mapping’ case, users will be asked for the information about the network environment if necessary. For example, the users will be first referred to the rules about checking the possibility of missing ‘PTR record’. Moreover, if the very mail servers are built on ADSL links, the cases might usually trigger additional processing. In these cases, because ADSL users usually have only parts of a CLASS C (i.e. 255 hosts) IP addresses, the PTR records of them usually have to be registered or configured through the related ISPs. Therefore, the users will be further referred to the rules in ‘DNS Registration’ KC.

Finally, if any of the problem cases has been identified, the final rule will trigger the ‘Suggestion’ KC to provide appropriate answer(s) for users to correct the problems as shown in Fig. 13.

5.3. Evaluation

To study the completeness of the system and to understand users’ acceptance, a questionnaire approach is adopted. We had invited a couple of domain experts and ordinary domain users to test the system. This questionnaire is built in the web page of the system, including the issues on correctness, acceptance, expressiveness, completeness, etc. Here is a simple summary.

- On the issue of correctness, we made requests for a couple of DNS experts to test our system. Thanks to their thorough examinations, some minor bugs had been identified and corrected in the first stage.
- On the other hand, for acceptance and expressiveness, most people acknowledge positive feedbacks on the adopted approach on our system. For example, some DNS beginners acknowledged that they could benefit...
much more from the system as compared to the traditional Q-n-A approach; however, if there could be more simple classification schemes and give more examples (e.g. from simple to advanced, in a hierarchical manner) on subsystems such as tutoring and term explanations, their acceptance will be higher.

- On the issue of completeness, it seems that there is still more to do for improving. While mail-related DNS problems are most concerned and hence are explored in much more details, other DNS problems such as DNS performance and security are still rather limited and need more efforts for improving on the issue of completeness.

Fig. 14 shows the daily statistics for September 2003 of our system. There are averagely 630 hits every day during September. In addition, we have a forum to collect the user feedback and bug reports.

6. Concluding remarks

In this paper, we design and implement an intelligent DNS Management System (iDNS-MS) and has shown that explicit DNS ontology can be used during knowledge engineering for dealing with the complexity of DNS knowledge management, especially for guiding the construction of the knowledge base. Our main contributions are: (1) to design and implement a unifying framework (e.g. including system file configuration, DNS-related problem diagnosis, planning, tutoring, etc.) for supporting intelligent DNS management using web interface and expert system technology; (2) to propose a ontology-driven model for eliciting rules from a previously built DNS ontology and constructing the objected-oriented knowledge base.

According to the experimental results, the paradigm of using DNS ontology to build iDNS-MS works good and effective. iDNS-MS will benefit the sharing and reusing of global DNS knowledge, the reduction of people’s time to learn DNS management, the ease of DNS configuration and planning, and the improvement on DNS and network operation. It is supposed that, the same approach could be adaptively modified to other problem domains for knowledge base construction.

Moreover, DNS maintenance is a sustained and evolving task that the functionalities and protocols of the DNS software might need to be updated from time to time. Future researches will focus on the following issues. We will extend iDNS-MS to integrate more topics including: the furtherance of IPv6, multi-lingual DNS, etc. In addition, 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 iDNS-MS with native language support (e.g. Chinese Big5, etc.) incorporated.

Acknowledgements

This work was supported in part by both the MOE Program of Excellence Research of the Republic of China
under Grant No. MOE89-E-FA-04-1-4 and National Science Council of the Republic of China under Grand No. NSC92-2213-E-009-099.

References


