基於本體論的DNS網路服務系統之研製

學生: 劉建良　　　　　　　　　　　　　　　　　指導教授: 曾憲雄
博士

國立交通大學電機資訊學院
資訊科學系

摘要

領域名稱伺服系統（Domain Name System，以下簡稱DNS）是現今網際網路基礎設施的重要環節之一。然而，目前市面上卻很少以 DNS 為主的專業網站。另外，隨著新的網路應用（比如說 IPv6 或是 ENUM)出現，DNS 的管理也變的更加複雜。我們在 2003 年提出一套智慧型DNS整合管理系統（iDNS-MS)的架構，並據以實作出一套實驗系統，開放於網路上面，提供給因為DNS系統管理問題而求助無門，以及其他希望了解更多 DNS 統合知識的網友，透過線上操作與學習，能夠學到更完整的 DNS 管理知識，得以掌握所管理系統的狀況，進而改善系統整體的服務效能。雖然 DNS 診斷系統可以提供使用者關於 DNS 問題所需要的建議，然而這樣的建議往往都是需要擁有相當的 DNS 背景知識才有辦法確切的瞭解。因此，與 DNS 診斷系統結合之 DNS 教學系統是非常需要的。除如何提供個人化的學習環境，規劃參考個人特質之適性化學習環境對於學生學習與教學教材的重複使用性與透通性也非常的重要；由於 SCORM 即可重複使用既有的教材，因此我們採用 SCORM 作為網路教學系統平台。另外除了診斷系統與教學系統之外，我們以 DNS 本體論架構作為背景知識，提出一個基於 DNS 本體論的三層式搜尋系統架構來輔助資料搜尋；三層式架構包含表現層、邏輯層與資料層，經由不同層次的分工可以讓整個架構更彈性並且擁有高度的可重複使用性。在本論文中，我們著重在使用專家系統與本體論技術來設計與實做包含診斷系統，教學系統與搜尋系的DNS 知識入口網站。

從 2003 年 iDNS-MS 開始對外提供服務而且收到的反應大都是正面的;所以基於 DNS 本體論來建構專家系統是可以得到不錯的成效的;透過整合診斷服務、教學服務與搜尋服務的 DNS 知識入口網站將可以把 DNS 知識做更大程度的分享與使用。另外,只要再稍加修改,同樣一套開發模式與所發展的技術,應該可以套用到許多科學及工程領域,來進行系統知識的擷取與類似專家系統的開發。

關鍵辭: 網域名稱伺服系統, 專家系統, 知識擷取, 知識本體, 規則擷取

Design and Implementation of Ontology-Based DNS Web Services

Student: Chien-Liang Liu Advisor: Dr. Shian-Shyong Tseng

Department of Computer and Information Science
National Chiao Tung University

ABSTRACT

The Domain Name System (DNS) is an essential part of the Internet infrastructure. However, few DNS professional web services can provide the DNS related knowledge. In addition, the new trend of DNS (such as IPv6 and ENUM) makes DNS management more complex. In 2003, we proposed a unifying intelligent system for DNS management, which provides the framework for DNS-related services. Although the diagnosis service could provide some suggestion about the DNS problem, for some novice DNS administrators, the suggested information is not enough. General speaking, the suggested information is not self-explanatory and often needs some DNS background knowledge to understand. Therefore, in addition to the online DNS diagnosis results, the DNS-related tutoring materials would also be required after the diagnosis process. In addition to the tutoring information combined with diagnosis system, tutoring system which could provide individualized learning environment, the reusability and interoperability issues of the teaching material are important as well. Since SCORM (Sharable Content Object Reference Model) could reuse existing teaching material, we adopt SCORM as web-based tutoring platform. In addition to diagnosis service and tutoring service, based on DNS ontology as the background knowledge, we propose a three-layer DNS ontology based search system framework to facilitate information search. The framework consists of presentation layer, logic layer and data layer. The separation of the layers would make the whole system more flexible and reusable. In this thesis, we focus on the design and building of ***DNS*** portal web service (including DNS diagnosis, DNS tutoring service and search service) by using knowledge-based system and ontological engineering technologies.

We have started to offer diagnosis service since 2003 and feedback shows that the paradigm of using DNS ontology to build knowledge-based system works good and effective. The integration of DNS diagnosis service, tutoring service and search service would benefits the sharing and reusing of DNS knowledge. In addition, with a few modifications, the same paradigm and developed algorithms could be easily adapted to other scientific or engineering domains.

Keywords: DNS, Knowledge-based System, Knowledge Acquisition, Ontology, Rule Extraction

誌謝

首先感謝恩師曾憲雄教授多年來的指導，沒有曾教授的指導，本論文將無法完成；在曾教授不厭其煩的指導下，引導我逐步學習與培養許多獨立研究的方法及技巧，讓我不僅僅只是完成本論文，同時也讓我瞭解到研究的本質；同時，也要感謝論文口試委員中央大學資工系 李允中教授、成功大學電機系 謝錫堃教授、東華大學電機系 趙涵捷教授、中央研究院資訊所 陳孟彰教授，以及本校資科系 孫春在教授、袁賢銘教授、施仁忠教授的建議，使整篇論文的理論與實務內容，更加趨於完整。

其次，本篇論文得以順利完成，要感謝許多人的協助、體諒與鼓勵。我必須要特別感謝陳昌盛博士以及陳瑞言、張俊彥兩位學弟的幫忙，使得本論文的主要理論基礎與架構，能夠逐漸成形，最後終於能圓滿完成；同時，也要感謝實驗室其他學長或學弟妹的幫忙，使得本篇論文實際的研究探索過程，能夠順利地進行，一直到論文完成。

最後，感謝家人的鼓勵與支持；在我決定念博士班的時候，你們的默默支持，讓我可以心無旁騖，專心於博士論文的研究；在我面臨挫折時，你們的關懷與支持，讓我能夠將眼前遭遇的困難，逐漸轉化成爲克服困境的動力，衝破難關，繼續堅持到底。

僅將這一篇論文，獻給老師、實驗室的學長與學弟妹、論文口試委員、以及我最親愛的家人。

Contents

List of Figures

List of Tables

List of Algorithms

Chapter 1 Introduction

The Domain Name System (DNS) is an essential part of the Internet software infrastructure. Unfortunately, due to the distributed nature of DNS systems and lack of efficient knowledge sharing mechanisms among DNS administrators, even though DNS is so important to network operation today, rather few DNS administrators have the expertise to do the jobs well. Besides, we could often find lots of poorly performed DNS servers on lots of Internet sites [M&M03]. In this thesis, we propose an ontology-based problem solving approach to strengthen the sharing of DNS knowledge.

Currently, most administrators learn to enhance their DNS management skills by fixing their encountered DNS problems or other reported cases through DNS administration books and public mailing lists such as those on ISC-BIND [BIND2005] web page. However, due to the limitation of one's own experience and lack of required domain knowledge about DNS, it is often the case that many people usually have a long and hard time before they could finally benefit from these readings and discussions. Moreover, with the furtherance of new DNS-related issues such as IPv6 [HD98], ENUM and multilingual DNS, the DNS management tasks might become even more complicated than ever before. Therefore, a system with integrated functionalities (including diagnosing, tutoring, etc.) to help DNS administrators learn and manage their DNS servers is required and highly recommended.

1.1 Motivation

C.S. Chen, S.S. Tseng, and C.L. Liu (2003) proposed a framework for the design and implementation of a unifying intelligent system (i.e., Integrated DNS

Management System, iDNS-MS) for DNS management, including DNS configuration, DNS design, outstanding traffic monitoring and analysis [CT+02-1], DNS diagnosis, and DNS tutoring systems [CT+03]. The iDNS-MS has started to provide services since 2003 and most of the feedbacks from users are positive. However, by analyzing the usage logs and studying the feedbacks collected, we find that there are still many ways for attacking the problem and improving. For example,

- First, although the diagnosis service could provide some suggestion about the DNS problem. However, for some novice DNS administrators, the suggested information is not enough. General speaking, the suggested information is not self-explanatory and it often needs some DNS background knowledge.
- Second, new Internet application issues should be incorporated into the existing system to enhance existing system.
- Third, the tutoring system is important for some DNS administrators, so the design of the learning sequence about DNS domain is important as well. With appropriate learning sequence design, the users will benefit more from the tutoring system.

In the following, we will briefly describe the main ideas of this research. First of all, even though the diagnosis service could provide the suggestions to network users, however, the provided suggestion information is not enough for many novice administrators. General speaking, the diagnosis suggestion focuses on how to fix users' problem only and it is often concise and pithy. In other words, the suggested information is not so self-explanatory and it needs some DNS background knowledge. However, for many novice DNS administrators, the incorrect configuration is due to that they do not have correct or enough DNS background knowledge. Therefore, in addition to the online DNS diagnosis results, the DNS-related tutoring materials would also be required after the diagnosis process.

On the other hand, in addition to the tutoring information combined with diagnosis system, the tutoring system is important as well. In iDNS-MS, we present the DNS tutoring system on the web using HTML format and topic-oriented structure. In general, the topic-oriented structure presents the teaching material passively and it could not present the most appropriate teaching material at appropriate time. General speaking, it is important to provide individualized learning environment and that would facilitate users' learning. Moreover, the reusability and interoperability issues of the teaching material are important as well. When the teaching material is reusable, other tutoring system could reuse the teaching material directly. On considering these, we adopt the SCORM (Sharable Content Object Reference Model) model for building the web-based tutoring system. Theoretically, SCORM is a suite of technical standards that enable web-based learning systems to find, import, share, reuse and export learning content in a standard way.

Moreover, new Internet services make DNS management more complex as well. For example, Internet telephony becomes more and more active now since Internet telephony system is motivated due to the possibility for cost-saving and the integration of new services. Nowadays, many people start consider about the possibility for the integration of voice and data applications that could connect the PSTN with the IP network and apply a unique identical methodology to provide most interesting services.

ENUM [Faltstrom00], developed as a solution to the question of how to find services on the Internet using only a telephone number, is the proposed IETF protocol that could assist in the convergence of the PSTN and the IP network. Since DNS is the existing distributed infrastructure for the translation between hostname and IP address and existing DNS system works well. Thus, ENUM propose to adopt DNS as ENUM infrastructure. An ENUM Domain Name System (DNS) [AL01] server is used to convert the phone numbers into the domain names and vice versa. In other words, it is the mapping of a telephone number from the PSTN to Internet services.

Furthermore, current Internet is mainly based on IPv4, which has shown its inability on adapting itself to many real-world applications. First, the shortage of IPv4 address space becomes a serious problem. For example, many telephony devices need the "always-on" [HH02] capability. In other words, these devices might need their own IP addresses during the communicating process. Second, new applications requiring important functionalities such as real-time and bandwidth reservation usually could not find good QoS (Quality of Service) support since IPv4 is primarily based on the best-effort working model. Third, the lack of data security and integrity mechanism on IPv4 becomes a big concern when e-commerce applications are performed on the Internet platform. Based on the above observations, IPv6 [HD98] [CH97], the next generation Internet protocol, is designed to replace IPv4. As we know, IPv6 having 128-bit IP address space not only could provide us enough IP addresses, but also could have a much better intrinsic security and QoS support. By these considerations, the IPv6 protocol stack is supposed to be required in ENUM environment and be superior to IPv4 for deploying massively IP telephony system. However, most people still have limited IPv6 experience. Hence, the dual-stack IPv4/IPv6 model is usually adopted by most sites as a solution.

In short, new application issues make DNS management more difficult. Therefore, it is supposed that a DNS portal system which could help novice DNS administrators learn and improve their DNS skills is required. In essence, our main contributions are listed as follows:

1. We propose an ontology-driven model for rules extraction. That could facilitate the rules extraction on DNS diagnosis system.
2. To eliminate self-explanatory problem of diagnosis system, we propose to

adopt model-tracing tutoring for further DNS tutoring. Besides, we propose an ontology-based model-tracing tutoring construction algorithm

3. We propose a DNS Ontology-Based search framework, which adopt DNS ontology as background knowledge, to facilitate the information search.
4. To reduce the complexity of SCORM learning sequence construction, we propose an ontology-based learning sequence construction model to generate the DNS learning sequence scheme.

1.2 DNS Ontology based Knowledge Portal

In this thesis, we pro pose to attack the above sub-problems by strengthening the iDNS-MS web services using DNS-portal like approach. Now the whole system consists of DNS diagnosis service, DNS tutoring service and DNS search service, where DNS diagnosis service helps DNS administrators diagnose their existing DNS servers, DNS tutoring service helps DNS administrators learn correct DNS knowledge, and DNS search service could help users search the information in the DNS portal system more efficiently. In essence, all the services are based on a DNS ontology. In DNS diagnosis service, we propose an ontology-driven rule extraction model to assist the rule generation. In tutoring service, we propose an ontology-based DNS model-tracing tutoring model to help knowledge engineer construct the skeleton of the model-tracing tutoring. In DNS search service, we make use of ontology concepts and relationships to enhance the search capability.

1.3 DNS diagnosis service

As with the popularity of Internet, the expert system (ES) [Durkin94] [Gaines00] 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 [NS+00].

In DNS diagnosis service, we adopt ***DRAMA/NORM*** [LT+03] 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 (Knowledge-Based System) for supporting intelligent DNS management through web interface. On the other hand, the knowledge model of ***DRARA/NORM*** is based on knowledge classes, which are like the concepts of ontology. Therefore, the transformation between the ontology concepts and the knowledge classes becomes 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 knowledge class and the relations between the knowledge classes 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, which facilitates the management of rules in each knowledge class. 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 thesis, we propose an ontology-driven model [LT+04-1] to help extract KBS rules from DNS problem cases. There are three phases in the ontology-driven model: ontology construction phase, knowledge class organization phase and facts/rules loading phase. Ontology construction phase is used to construct the domain ontology, knowledge class organization phase is used to organize the relationship between the knowledge classes, and facts/rules loading phase is used to fill in the facts/rules of knowledge classes extracted from domain experts. As mentioned in [CJ+99], 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.

1.4 Ontology-based DNS Model-Tracing Tutoring System

Currently, with the exception of some specific applications (e.g., peer-to-peer applications [Shirky00], etc.), most internetworking services are based on the working model in which there will be some successful DNS queries before the communication activities. In principle, the hierarchical and distributed properties of the DNS system make the administration duties to be distributed among different organizations and networking sites and make the whole system more scalable and robust. However, the debugging and tracing issues of network system become more difficult as well. Many network services might not work properly and seem to fail whenever there are contingency events that make their DNS servers unable to work properly as expected.

Theoretically, DNS tutoring service is important for those who would like to know the DNS operation principles in more detail. If DNS administrators could have basic and correct DNS knowledge, the possibility of incorrect configuration would be less when building a new DNS service. Thus, we design and implement a DNS model-tracing tutoring system. Instead of providing DNS diagnosis and tutoring course separately, we further propose a diagnosis-driven tutoring system to address these kinds of issues. In practice, through DNS diagnosis process, users could identify their problems and the DNS configuration information reflects the users' activities on DNS server. In essence, DNS diagnosis system could be viewed as problem-driven model and diagnosis rules could be used to trace users' action. In theory, model-tracing methodology [AB+90] for tutoring is based on the ACT theory of skill acquisition. Accordingly, a skill can be analyzed into a set of productions rules and instruction can be organized around these rules. Based on above observations, it seems model-tracing tutoring is appropriate for DNS tutoring system.

However, model-tracing tutoring construction is not necessarily a straightforward job; it usually needs some knowledge acquisition to help construct the model. Consequently, we further propose an ontology-based approach for the model-tracing tutoring skeleton construction [LT04-3]. The main functionality of ontology-based model-tracing model is to help the knowledge engineers construct the skeleton of model-tracing tutoring with the help of ontology and extract the production rules for simulating users' behaviors.

1.5 DNS Ontology-based Search Engine

Search engine often plays an important role in the information system or portal server. Because much information exists in the Internet or system, search engine is one of the most convenient tools for us to find required information. However, most of traditional search engines are based on keyword search which ignores semantic information. The drawbacks of keyword search are listed as follows:

1. Ambiguity problems:

Term ambiguity often occurs during keyword search and that would lead to irrelevant information result. For example, the single term “bank” could be referred to the institution that accepts money deposits or the slope beside a body of water. Without any other information providing, the search engine would misunderstand the meaning.

2. Expression problems

Sometimes it is not easy to express what we want with keyword expression. Especially when we are not familiar with that domain, general term expression would be a convenient way. For example, for most of novice DNS administrators, they know that the term “DNS security” could be used to represent DNS security issues, but they do not know the specific DNS security issues (e.g. “DNS Dynamic Update” or “Zone Data Protection” issues).

3. Synonym problem

Different domains have domain-specific abbreviation about the term sometimes. For example, in DNS domain, the term “Master DNS” is identical to “Primary DNS”. Without the background knowledge, the users would miss some required information. When users would like to search “Master DNS” information and enter “Master DNS” as the keyword, “SPOF” information may be excluded.

In essence, the domain ontology could represent the term semantics by the concepts and relationships between the concepts. Besides, if the application focuses on specific domain, ontology would be viewed as the background knowledge of the domain and that would improve the search capability. Hence, based on DNS ontology as the background knowledge, we propose a three-layer DNS ontology based search system framework, which consists of presentation layer, logic layer and data layer. The separation of the layers would make the whole system more flexible and reusable. With minor modification, we could change the presentation from web interface to other user interfaces (e.g. PDA, email, etc.). Moreover, the flexibility of importing new data source (e.g. mailing list archie, PDF files, WORD files, etc.) is reserved.

Chapter 2   Preliminaries

In our system design, we adopt expert system as the system backend system. Therefore, we need to perform knowledge acquisition process to extract knowledge from domain experts. General speaking, different knowledge representation schemes exist for the knowledge representation. Different knowledge representations have different focus. In our system, we adopt ontology and rules as the knowledge representation schemes. In C.S. Chen, S.S. Tseng and C.L. Liu (2003), the DNS ontology is constructed based on the use case modeling. The middle-out approach takes into account the cases from users and the skeleton structure from domain experts both and then perform the merge process to combine these two kinds of DNS knowledge. Use case modeling for ontology construction works well and could make the knowledge acquisition process more successfully. In essence, the ontology should be able to evolve when the original knowledge modified or new knowledge comes. In Chen et al. (2003), the DNS ontology focuses on IPv4 only and that needs some modification because of the requirement of new applications domain.

In addition to diagnosis system, the intelligent tutoring system is important as well. The integration of diagnosis system and tutoring system would be helpful for those who would like to know the DNS operation model more detail after the diagnosis process. Furthermore, we adopt SCROM standard as the web-based learning platform to achieve the goal of reusability and interoperability. In Section 2.1, we would describe the DNS domain knowledge and ontology representation. In Section 2.2, we would describe the new application trend that related to DNS. Section 2.3 and Section 2.4 introduce model-tracing tutoring and SCORM, respectively. Finally, in Section 2.5, we describe the DRAMA/NORM, which is the expert system shell in our system design.

2.1 DNS Domain Knowledge and Ontology

2.1.1 Basics of the DNS System

The Domain Name System [Mockapetris87-1, Mockapetris87-2] is responsible for translating between hostnames and the corresponding IP addresses needed by software. The mapping of data is stored in a tree-structured distributed database where each name server is authoritative (responsible) for a portion of the naming hierarchy tree. The client side query process typically starts with an application program on the end user's workstation, which contacts a local name server via a resolver library. That client side name server queries the root servers for the name in question and gets back a referral to a name server who should know the answer. The client's name server will recursively follow referrals re-asking the query until it gets an answer or is told there is none. Caching of that answer should happen at all name servers except those at the root or top-level domains (.com for example). The working paradigm could be illustrated in Fig. 2.1.

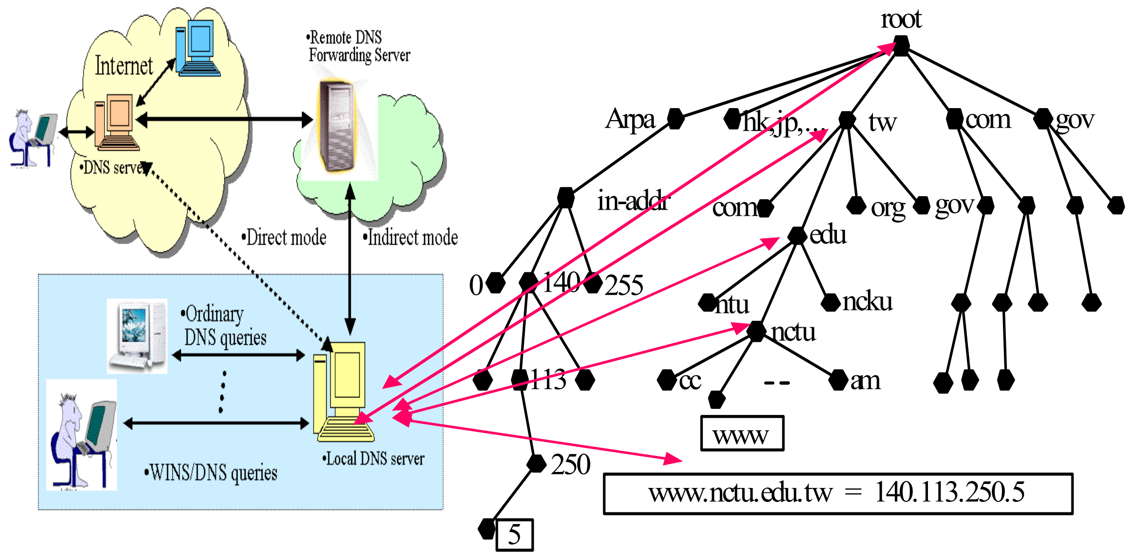


Fig. 2.1: DNS operation model

There are many operational, planning and management issues that need expertise to improve the DNS system. Unfortunately, new administrators or administrators that manage a small scale of network usually do not know the theoretical and practical knowledge of DNS system very well. It takes a long time for them to gain the related knowledge without the assistance of the experts.

Table 2.1: A simple classification of typical DNS problems

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

Table 2.1 shows a simple classification of DNS problems that most DNS

administrators might encounter. Due to the complex and distributed nature of the DNS system, we could often find lots of poorly performed DNS servers (i.e. by mis-configuration, inappropriate planning, etc.) on lots of Internet sites. Many factors contribute to these and the important ones are listed below:

- 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 the experts.
- Many administrators that manage a small scale of network lack the experiences for 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 seem immaterial on the sites; however, these issues can become fatal problems when the overall traffic grows larger and larger.
- Moreover, given the importance of DNS servers, direct or indirect attacks on the DNS systems are common [BIND05] [Hanley00] [Koh01]. 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 a beginning of a new wave of attacks against vulnerable DNS server infrastructures.

As mentioned in [CT+02-2], many companies and people develop assistant software to help DNS administrators managing their DNS systems as shown in Table 2.2. However, most of these software packages are built by using conventional methodology. Basically, they are mainly used to solve syntax problems and provide friendly user 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. DNSreport [DNSreport05], which is popular now, provides a web site to help DNS administrators to find DNS problems and to fix them. All users need to do is enter a domain name that they want, and this site will report DNS problems. However, this report lacks：

- For DNS beginners, this report will be useless if there is no DNS server.
- For DNS planning, such as Topology, DNS performance, and DNS security, this report can not provide any suggestion.
- There is no debugging function for DNS configuration in DNSreport.
- DNSreport lacks detailed knowledge for users to learn how the problems occurred or how to avoid similar problems.

Table 2.2: List of DNS assistant software

Software	Benefits	Company
Quick DNS	■ Manages more zones in less time (i.e., time saving zone editor). ■ Manages larger zones in less time (i.e., automatic set-up of secondary DNS servers). ■ Manages DNS while at home or on the road (i.e., fast remote management).	Men & Mice
DNS Expert AD	■ Bridges the gap between active directory and DNS. ■ Helps prevent active directory errors. ■ Reports on 200 DNS and AD configuration errors.	Men & Mice
DNS Expert Monitor	■ Warns instantly of errors and helps users fix them. ■ Saves valuable troubleshooting and maintenance time. ■ Monitors internal and external DNS on any platform.	Men & Mice

	■ Increases security level from malicious attacks.	
DNS Expert	■ Verifies DNS setup for reliability. ■ Tests for availability of backup mail and DNS services. ■ Checks for the general configuration of zones and connections to the parent domain(s). ■ Conducts security tests for DNS spoofing and mail rely.	Men & Mice
Dlint	■ Conducts DNS Server Zone verification. ■ Analyzes DNS zone. ■ Reports zone problems.	Domtools
DOMTOOLS	■ Provides some high-level tools that do things, which most DNS administrators will find valuable. ■ Provides computer-parsable output from all commands so that high-level tools are easy to develop.	Domtools
DNSstuff	■ Provides many web-based tools to verify network conditions.	DNSstuff

2.1.2 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 [CJ+99]. 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 [HS+97]. 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 [SO+00].

As shown in Fig. 2.2 [CT+03], we extract the concepts and attributes by using a hybrid method consisting of the brainstorming and use case modeling [Cockburn97]. 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.

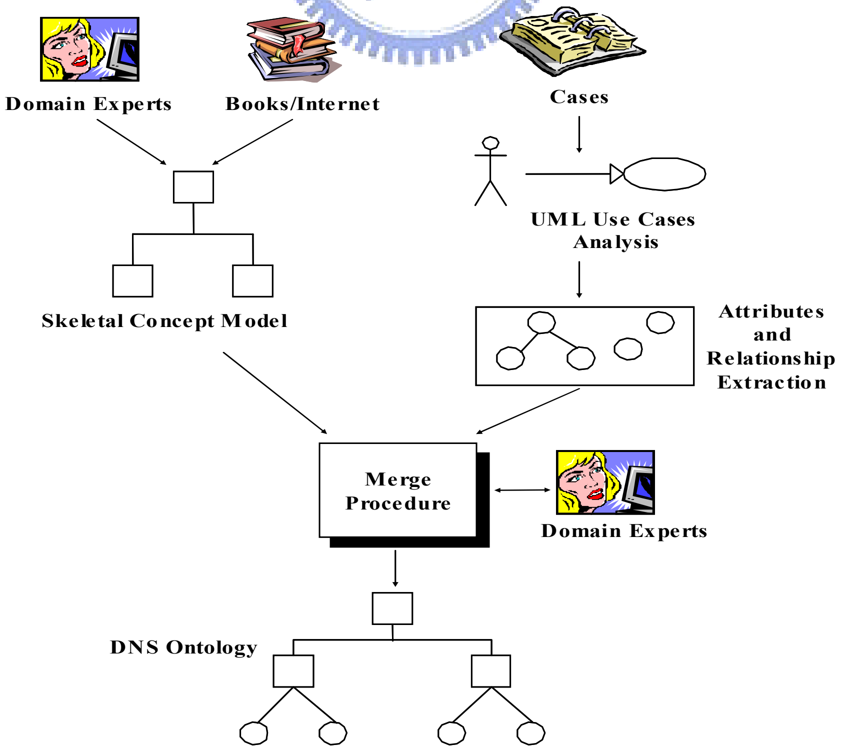


Fig. 2.2: DNS modeling and DNS ontology construction

2.1.3 DNS 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: ***hostId*** (i.e., IPv4/IPv6 address), ***serverType*** (e.g., authoritative server, caching server, etc.), ***hostInventory*** (e.g., 1Gb RAM, 2.80-GHz CPU, 100Mb 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 SPOF (Single-Point-Of-Failure) and DNS configuration error (e.g., lamed DNS servers) are typical types of the DNS availability problems. Hence, the SPOF concept (and lame-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.

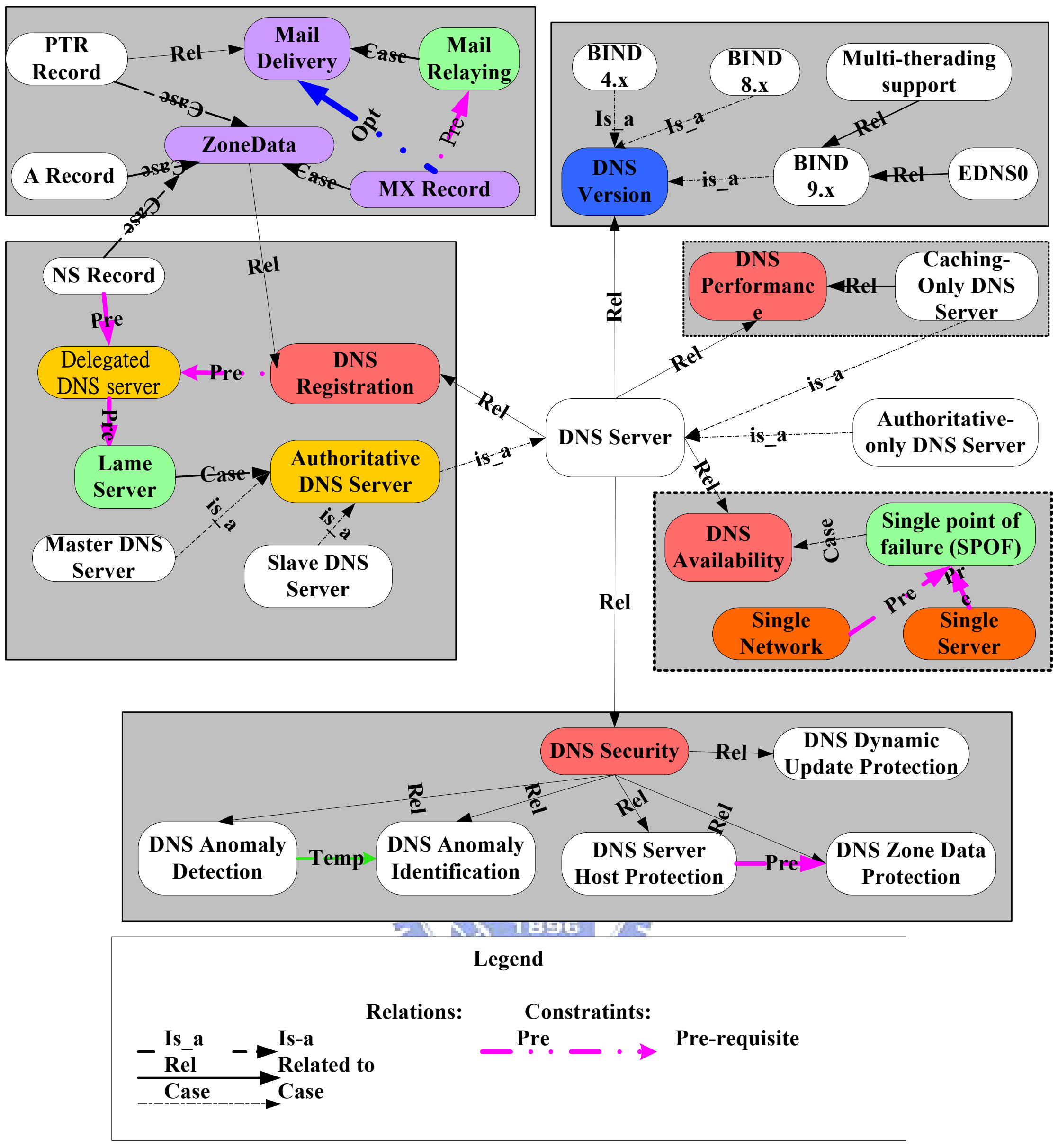


Fig. 2.3: Snapshot of DNS ontology

In essence, ontology representation is suitable for communications and natural for human thinking, meanwhile rule representation is powerful for machine to manipulate the concepts. 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 rule-base 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. 2.3 shows a snapshot of DNS ontology [CT+02-2]. 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, "Single Point of Failure (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 (Single-Point-Of-Failure)" concept depends on many concepts including: "Single Network", "Single Router" and "Single Server".

2.2 New Trend in DNS

2.2.1 DNS and IPv6

The DNS is an essential part of the Internet infrastructure since it provides not only an efficient and distributed working model, but also a universal global addressing mechanism [AL01]. We need the help of DNS to translate the domain names into IP addresses and vice versa. This is especially true on IPv6 environment, since the 128-bit address makes it difficult for most people to remember.

Moreover, in the process of migration from IPv4 into IPv6, or running in a hybrid IPv4/IPv6 environment, the administrators have to do a lot of things. First, almost all applications need updating to support both IPv4 and IPv6. For example, if e-mail routing, including both IPv4 and IPv6, is inappropriately configured, mails might not be delivered successfully to their destinations, or even might get lost in IPv6/IPv4 environment. Second, the DNS server programs also have to support both protocol stacks as well. Finally, since there are inherently different management issues between IPv4 and IPv6 DNS, the adjustment of the DNS should be adaptive. For example, in the DNS, only the IPv4 address records (e.g., ***A*** and ***PTR***), or the IPv6 address records (e.g. ***AAAA***, ***A6*** and ***PTR***), or both groups of IPv4/IPv6 records can be stored for each name. In the last case, deciding whether to use the IPv4 or IPv6 address is not easy, and the choice is the result of much consideration. At first, determining whether the node has an IPv6 direct connectivity is necessary. If not, the use of the IPv6 address will require the transmission of an IPv6 packet in an IPv4 tunnel. This approach can be less convenient than the use of native IPv4 or even impossible if the node cannot use tunnels.

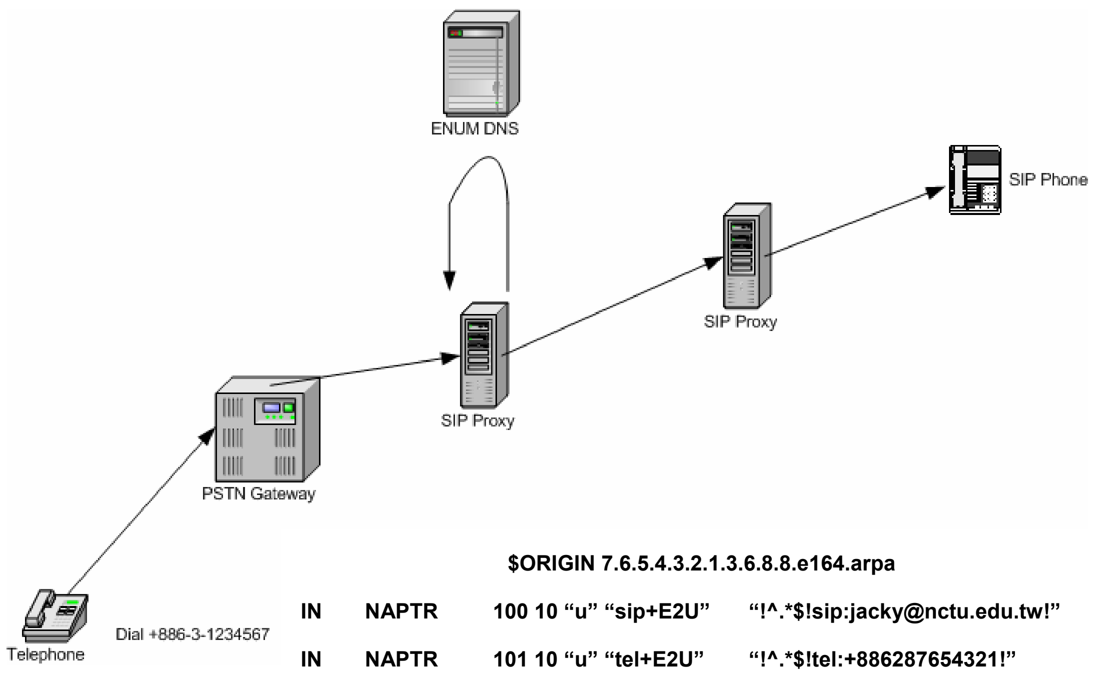
2.2.2 SIP and ENUM

SIP (Session Initiation Protocol) [HS+99] is a signaling protocol for Internet multimedia conferencing, Internet telephone calls and multimedia distribution. SIP supports five features of establishing and terminating multimedia communications, user location, user availability, user capabilities, session setup and session management. User location, user availability and user capabilities indicate where the callee is, whether the callee is available or not and what kind of service the callee accepts respectively. SIP invitations used to create sessions carry session descriptions, which allow participants to agree on a set of compatible media types. When the user would like to send a request, the request will be sent to a locally configured SIP proxy server or to the corresponding IP address and port according to the request-URI.

For example, SIP applications could not only connect the IP system, but also work with traditional PSTN telephone system. With the help of SIP/PSTN gateways, the SIP clients could reach PSTN clients and vice versa. There are three types of SIP servers, namely, SIP proxy servers, SIP redirect servers and SIP registrar servers. A SIP proxy server forwards requests from user agents to next SIP servers. A SIP redirect server responds to client requests and informs them of the requested servers' addresses. A SIP register server receives registration information from user agents and saves them in a location service using a non-SIP protocol and informs the user agents.

To achieve the above functionalities, SIP applications need a global addressing mechanism; that is, each client needs a unique identify address for facilitating the locating of the corresponding caller easily. There are a number of possible candidates such as E-mail address and telephone numbers for implementing the unique identifying mechanism. In particular, telephone numbers are preferred for most PSTN clients since the installed base is much bigger than email addresses and they are easier to remember.

One of the primary goals of ENUM is that each user can be reached in a number of ways by using only one number. To accomplish this, we need some mechanisms to make the phone numbers globally accessible and the subscribers can define their preferences for incoming communications. As mentioned previously, the DNS provides not only an efficient and distributed working model, but also a universal global addressing mechanism. Therefore, it is appropriate to choose the DNS for implementing ENUM. For example, based on [Faltstrom00], the phone number +886-3-1234567 will be converted into the domain name ***7.6.5.4.3.2.1.3.6.8.8.e164.arpa.*** The NAPTR [MD00] record could be used for identifying available ways of contacting a specific node identified by that name. Specifically, it can be used for finding out what services exist for a specific domain name, including phone numbers by the use of the ***e164.arpa*** domain. As shown in Figure 2.4, when the user dials the telephone number, the number will be translated into the corresponding domain name. Just like general domain name queries, the DNS server will return the related NAPTR records for this domain name. In this case, based on the NAPTR information, we could find out that there are two kinds of contacting methods and SIP protocol is the preferred method.



```
$ORIGIN 7.6.5.4.3.2.1.3.6.8.8.e164.arpa
IN   NAPTR   100 10 “u” “sip+E2U”   “!^.*$!sip:jacky@nctu.edu.tw!”
IN   NAPTR   101 10 “u” “tel+E2U”   “!^.*$!tel:+886287654321!”
```

Fig. 2.4: ENUM operational model

2.3 Model Tracing Tutoring

In traditional classroom instruction approach, it is not easy for students to receive one-on-one instruction. The concept, known as intelligent tutoring systems (ITS) or intelligent computer-aided instruction (ICAI), has been pursued for more than three decades by researchers in education, psychology, and artificial intelligence. The goal of ITS is to provide individualized tutoring automatically and cost-effectively. To achieve the goal, ITS needs to consider what students know, what the students need to know and which part of the curriculum is to be taught next.

Model-tracing methodology [AB+90] for tutoring is based on the ACT [AC93] theory of skill acquisition. According to the theory, a skill can be decomposed into a set of productions rules and instructions can be organized around these rules. Students’ problem-solving behavior can be interpreted and tutored by tracing their solution through production rules [AP91]. Model-tracing tutoring has been successfully used on tutors for many domains (e.g. LISP programming, high-school geometry and algebraic manipulation etc.). Instead of telling the students the correct answers directly, model-tracing tutors try to simulate users' activities by the production rules and provide appropriate assistances when needed. Even though the users enter incorrect answers, we could still get some information from their answers. For example, as shown in Fig. 2.5, there is an example algebra equation "$3-3(x-4)=-x$" and if we represent all possible problem-solving answers (correct and incorrect) with tree nodes and have them connected, the whole problem-solving space could be represented by using tree structure. As we know, one of the correct problem-solving paths in Fig. 2.4 could be derived from the path "node1-node3-node5-node7-node8". However, in practice, many students may derive incorrect answers from some alternative paths for the sample problem. Hence, it is important that the system should (or could) provide appropriate assistances or online help when users enter incorrect paths. For example, when the users go through the path, "node1-node3-node5-node6", we could infer that they have made the classical sign error. The tutors should recognize this kind of errors and provide appropriate remedial message if users request for help.

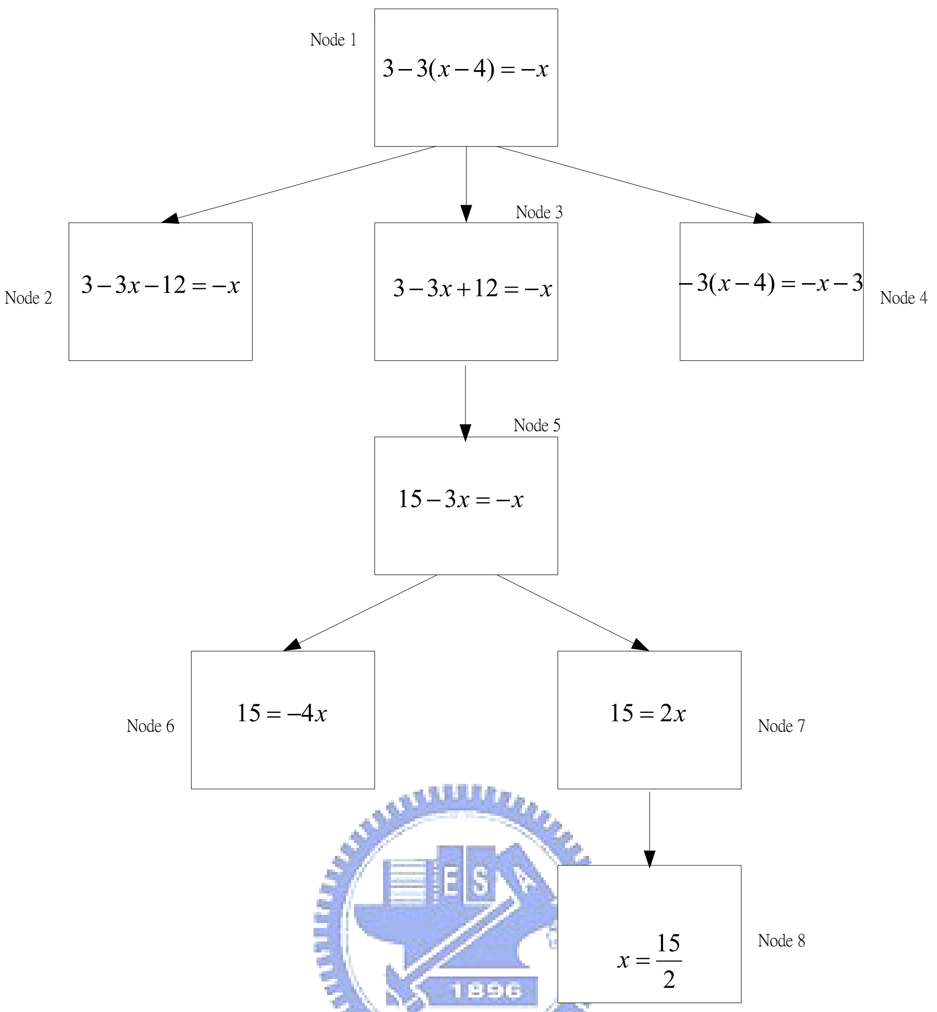


Fig. 2.5: Possible problem-solving path about algebra problem

Even though model-tracing tutoring has been shown to be a promising approach for building educational systems, yet the process of building model-tracing tutor is not easy. For example, as described above, for a simple equation like, $3-3(x-3)=-x$, there might be lots of possible problem-solving paths. Nonetheless, the more possible paths are found, the more information about users' activities is obtained. Therefore, if we would like to design production rules for specific problem domain, there must be some mechanisms for KEs to decompose the problems into sub-problems and analyze users' activities against the production rules.

2.4 Sharable Content Object Reference Model (SCORM)

In recent years, many e-learning standards have been developed. The Sharable Content Object Reference Model (SCORM) is an aggregated specification for asynchronous distance learning, organized by the Advanced Distributed Learning Initiative (ADL) (http://www.adlnet.org/). SCORM contains the definitions about the meta-data of learning material, Content Aggregation Model (CAM) which defines how to organize a course into a tree-like structure called Activity Tree (AT). Fig. 2.6 shows an example of AT. It is a structure that provides the hierarchical organization of learning content. According to SCORM 1.3 specification, an AT is structured by a set of clusters. A cluster is an organized aggregation of activities consisting of a single parent activity and its first level children, but not the descendants of its children. The cluster is considered to be the basic sequencing building block. The parent activity of a cluster will contain the information about the sequencing strategy for the cluster. The status information of all child activities will be collected and can be used to sequence these activities in the structure.

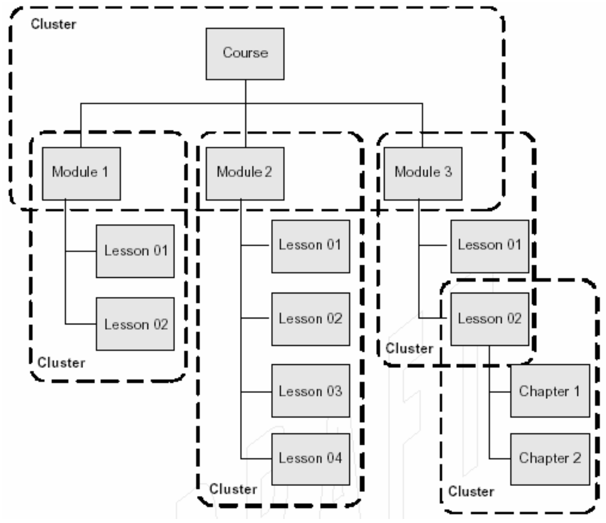


Fig. 2.6: An activity tree with clusters

2.4.1 The Sequencing and Navigation (SN) Specification

The *SCORM Sequencing & Navigation* (*SN*) *Specification* is based upon the Instructional Management System (IMS, http://www.imsproject.org/) Simple Sequencing Definition Model. It provides a profile about information of specific behaviors between activities and restrictions while learning an activity. The *Sequencing Definition Model* (SDM) defines the following categories: *Sequencing Control Modes, Sequencing Rules, Limit Conditions, Auxiliary Resource, Objectives, Objective Map, Rollup Controls, Selection Controls, Randomization Controls* and *Delivery Controls*.

(1) **Sequencing Control Mode (SCM):**

The Sequencing Control Mode (SCM) allows the content developer to determine how navigation requests are applied to a cluster and how the cluster's activities are considered while processing sequencing requests. Table 2.3 describes the SCM that may be applied. Sequencing Control Modes can be applied to any activity in the AT and multiple modes are enabled to create combination of control mode behaviors. Nevertheless, the *Sequencing Control Choice*, *Sequencing Control Flow* and *Sequencing Control Forward Only* modes will have no effect if applied to leaf activities.

Table 2.3: The description of Sequencing Control Mode (SCM)

SDM	Description
Sequencing Control Choice	Indicates that a Choice navigation request is permitted to target the children of the activity

Sequencing Control Choice Exit	Indicates that the activity is permitted to terminate if a Choice sequencing request is processed.
Sequencing Control Flow	Indicates the Flow Sub-process may be applied to the children of the activity.
Sequencing Control Forward Only	Indicates that backward targets (in terms of Activity Tree traversal) are not permitted for the children of the activity.
Use Current Attempt Objective Information	Indicates that the Objective Progress Information for the children of the activity will only be used in rule evaluations and rollup if that information was recorded during the current attempt on the activity.
Use Current Attempt Progress Information	Indicates that the Attempt Progress Information for the children of the activity will only be used in rule evaluations and rollup if that information was recorded during the current attempt on the activity.

(2) Sequencing Rule:

The IMS Simple Sequencing Specification (IMS SSS) employs a rule-based sequencing model. The behaviors between activities are defined by Sequencing Rules. Sequencing Rule is composed of a set of conditions and a corresponding action. The structure of sequencing rule is:

if *[condition_set]* ***then*** *[action].*

The conditions are evaluated using tracking information with the activity. The action of sequencing rule will be triggered if its condition-set evaluates to true. There are three kinds of sequencing actions SCORM proposes: ***Precondition Actions, Post-condition Actions*** and ***Exit Actions***, which describe different learning strategies.

(3) Objective:

IMS SSS proposes a mechanism of objectives of each activity for sequencing propose. Each learning objective associated with an activity will have a set of tracking status information which is used to decide which sequencing decision should be triggered according to student's current learning progress. Two kinds of learning objective are defined in IMS SSS: ***Local Objective*** and ***Global Shared Objective***. The Local Objective is only referenced by one activity; however, the Global Shared Objective can be shared by sets of activities. Therefore, activities may have more than one associated local objective and may reference multiple global shared objectives. Fig. 2.7 shows an example of objectives. All objectives except Objective 5 are local to their associated activities; Objective 5 is a global shared objective shared between Activity **AA** and Activity **BB**.

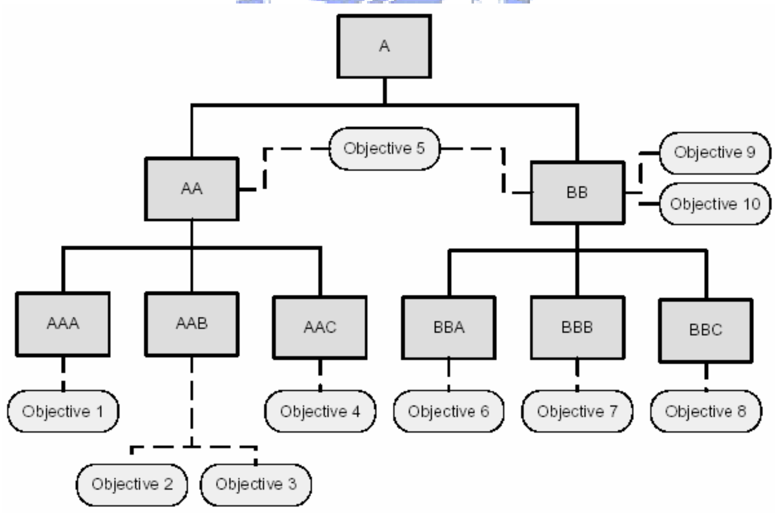


Fig. 2.7: An example of objectives

(4) Rollup Rule

Cluster activities are not associated with teaching materials; therefore, there is no direct way for learner progress information to be applied to a cluster activity. The

IMS SSS defines the way of how to evaluate the learner progress of cluster activity. The structure of rollup rule is:

if *[condition_set] True* ***for*** *[child activity set]* ***then*** *[action].*

The conditions of rollup rule are evaluated against the tracking information of the included child activities, and a corresponding action will set the cluster's tracking status information if the conditions are evaluated to true.

2.4.2 Tracking Model

The tracking model is a collection of dynamic sequencing state information associated with each activity in the activity tree for each learner. Tracking model elements will be updated to reflect learner interactions with the currently launched content object during a learning experience. It defines the following sets of tracking status information:

(1) Objective Progress Information: describe the learner's progress related to a learning objective.

(2) Activity Progress Information: describe a learner's progress on an activity. This information describes the cumulative learner progress across all attempts on an activity.

(3) Attempt Progress Information: describe a learner's progress on an activity. This information describes the attempted progress on an activity. Fig. 2.7 shows the Tracking Models for an Activity Tree.

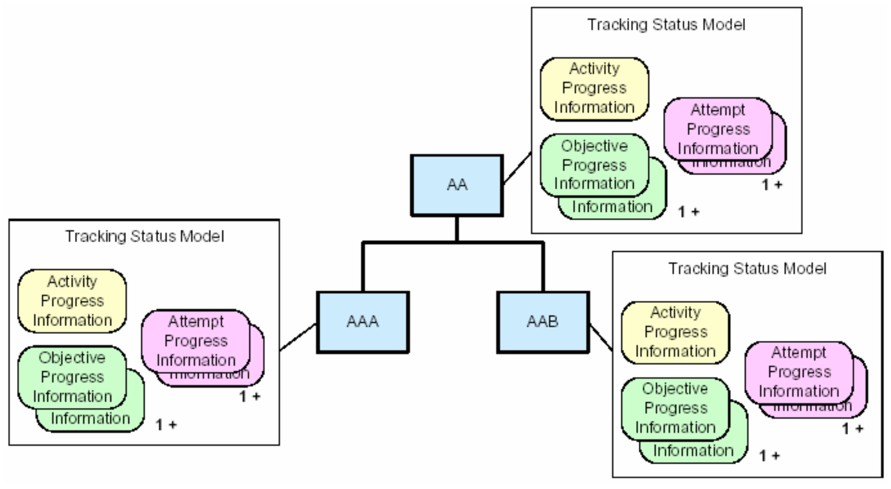


Fig. 2.8: The tracking models

Currently, more and more researches about constructing an intelligent tutoring system based on SCORM standard. However, the processes of building an activity tree and defining sequencing behaviors are very complicated for teachers, because the formats of meta-data and Simple Sequencing are described by XML. The functionality within a lesson or between lessons is hard-coded whether based on linear or an adaptive model. It means teachers must edit lots of XML files for building a course; definitely, it will bring more burdens to teachers and limit the reusability of individual learning objects (SCOs). It also limits the ability to create new or custom content structures from the same instructional materials. Therefore, in addition to the tools for editing the SCORM-compatible content packages, the mechanism for the SCORM learning sequence construction is important as well.

2.5 Overview of DRAMA/NORM

In traditional forward rule-base expert system, the rule base consists of all rules and facts. The system needs to go through every matching rule when conducting inference for the proper result. 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.

We apply the DRAMA/NORM package for building up the expert system. DRAMA is a rule-based, client-server tool/environment for KBS development. It can assist knowledge engineers in building up an expert system. Briefly, DRAMA contains lots of innovative techniques including Object-Oriented technology, knowledge inheritance, etc. It also contains useful tools, like rule verification tool, knowledge acquisition assistant tool and the inference server. Using the client-server architecture of DRAMA, the knowledge base is maintained on a server and clients could access this server for inference services.

The kernel knowledge model of DRAMA, named NORM (New Object-Oriented Rule-base Model), is developed by the KDE Lab at Dept. of Computer & Information Science of National Chiao-Tung University. The working model of NORM, containing knowledge classes (KCs) and the relationships between KCs, is based on the principles about how people ponder and learn to acquire knowledge.

According to domain expertise, when a person is trying to learn something, there are often some topics for him/her to study. A lot of new knowledge is built upon the original knowledge according to the discipline of Educational Psychology. Thus, new knowledge about the topics could easily be built one by one after the person successfully studies them. And, these topics could be transformed to KCs easily. In other words, learning is an activity to construct the relationships between different KCs. Since this knowledge model fits in quite well with the thought of human and KCs are modularized, we can build and maintain the knowledge base more conveniently. It is very important to use such knowledge model for the knowledge engineers. Whenever there is a need to update some knowledge, it is unnecessary to change all the knowledge base. All we have to do is just to add or modify the modules involved. In addition, the client-server architecture of DRAMA makes the web services plausible and more easily. Thus, the benefits of the expert system approach can be utilized throughout the Internet.

Chapter 3 The Problem Situations

3.1 Knowledge-Based System Rules Extraction

In 2003, we design and implement the DNS diagnosis system which could tell the DNS administrators if their system(s) work as expected. The diagnosis system has opened to the public since 2003 and the diagnosis model for DNS domain works wells and most of the feedbacks are positive. In practice, the KBS should be able to evolve as well. In other words, when the new knowledge is discovered or the old knowledge should be modified, the KEs would update the KBS. Since our diagnosis system is rule based knowledge system, the new knowledge means new rules should be discovered.

However, the rules extraction is not necessary a straightforward job. In general, the knowledge engineers are not familiar with the domain related knowledge, while the domain experts do not know how to express their own knowledge explicitly. The KA problem often dominates KBS construction process among the problems and resembles the system analysis in the same way as the expert systems resemble the classical computer programs. The problems that we are faced with during the KA process are usually very hard. 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) creation of a model of the expert's domain knowledge and performance.

The KA process is not a monolithic process but makes use of many sources of information in several forms, such as specifications, experience, principles, laws, observation, and so on, recorded in a variety of media. Knowledge sources are where we get data that related to our problem domain. After data collection, we use the knowledge acquisition method to transfer the data into knowledge. In this work, there are three main knowledge sources and we will describe each of them briefly in the following.

■ Domain Experts

Experts are those who have domain knowledge that can help knowledge engineers to understand more about the problem domain and find appropriate ways to construct and represent the domain knowledge. However, since not all experts could show their expertise, knowledge engineers must learn some communication skills to help get the required information from the domain experts. In this work, we have interviewed several human experts that mastered the skills in the DNS management and planning over years.

■ Documents

Documents are another important type of knowledge sources. Before the knowledge engineers interview the experts, they have to read some documents to help themselves understand the basics of the problem domain. Furthermore, these documents can also provide the KE's with some general ideas, such as how to divide the entire problem into sub problems or what kind of attributes are more important and more relevant to the problem domain. So when they interview the experts, they can ask more proper and advanced questions to acquire more knowledge from them.

■ Experiment results

Since the DNS system is such an important system to the network infrastructure, some groups had performed a lot of experiments to evaluate the effects when they applied some management strategies to the DNS servers. The results and the strategies of these experiments also provide us with some insight and important issues for developing our system.

3.2 Intelligent Tutoring System

In traditional tutoring system, teaching materials are organized by chapters and students usually learn the topics sequentially. In general, however, the chapter-structure representation of DNS domain knowledge might not be a good enough way for many people (i.e., especially for the inexperience DNS administrators) on DNS learning for several reasons. First, when dealing with abrupt DNS problems, many inexperienced administrators would like to know the reasons leading to the problems and how to fix them quickly instead of learning all the DNS-related knowledge sequentially. Second, many internetworking problems, looking like unrelated to DNS at first, happened due to improper configuration or deployment of the DNS systems. The typical ones include: (1) not knowing how to configure the DNS MX Resource Records for deploying multiple mail gateways (i.e., to facilitate the anti-spam and anti-virus filtering on the mail system); (2) not knowing how to protect an authoritative DNS server (e.g., a master or slave server) of the specified zone from abusing; and (3) not knowing how to avoid DNS SPOF problems (i.e., DNS-SPOF might affect the overall internetworking operation of the site severely under specific environment).

Therefore, it is supposed that the problem-driven approach is a more appropriate way for DNS tutoring than the traditional one. Since the original DNS diagnosis subsystem focuses on the DNS problems only, this approach might fail to address the needs of some inexperience people. On considering these, we propose to refine our DNS tutoring system with model-tracing theory and have it integrated with the DNS diagnosis subsystem. As compared to the traditional tutoring approach, it is supposed that most users could benefit much more from the refined DNS tutoring.

3.3 Intelligent Search System

In the search system, we often adopt the keyword search as the front-end. In addition to the keywords mechanism, some search system would provide the ***Boolean*** operations to enhance the search capability. However, most of the search systems focusing on the keywords only may lead to information loss. The semantics of the terms could make the search system more intelligent. In addition, if we would like to rely on the agents for the search system, the semantics of the term would be very important. In general, to achieve the goal of intelligent search system, there are two approaches.

1. Make the data source and the query terms both semantic and apply the semantic query string on the semantic data source. To make the data source semantic is not an easy job, since we have a common vocabulary to communicate. In practice, W3C proposes to use semantic web to achieve the goal of information exchange between human and machines. The Semantic Web is an extension of the current web in which information is given well-defined meaning, better enabling computers and people to work in cooperation (Berners-Lee et al., 2001). It is a collaborative effort led by W3C with participation from a large number of researchers and industrial partners. It is based on the Resource Description Framework (RDF), which integrates a variety of applications using XML for syntax and URIs for naming. However, for most of the people, RDF is more complex than HTML and most of the web pages are still semantics-less.
2. Make the query terms be semantic only and apply the semantic query string on the original data source. If we focus on specific domain, the domain ontology would be more easily built and we could transform the query string into semantic

ones. In theory, this approach is less effective. However, this approach is more practical in existing environment.

3.4 Typical DNS management issues

Table 3.1: Management issues of IPv4, IPv6 and ENUM DNS

Item	Descriptions	IPv4	IPv6	ENUM
Correctness (Configuration)	Delegations of domain zones, illegal setting of DNS entries, etc.	●	●	● ●
Availability	Master/slave architecture, data synchronization among authoritative servers, etc.	●	●	● ●
Performance	DNS caching, forwarding, etc.	●	●	● ●
Security	Access control, Dynamic Update, Intrusion detection, etc.	●	●	● ●
Software Interoperability	BIND (version 4,8,9,etc.), Microsoft DNS, etc.	●	●	●
IPv6/IPv4 Interoperability	IPv6, IPv4		●	●
512 bytes limit in DNS query/answer UDP packet	Some of the older DNS server software could not transfer the packet with TCP when query/answer UDP packet is larger than 512 bytes		●	●

Table 3.1 shows typical DNS management issues concerning IPv4, IPv6 and ENUM DNS. The correctness issues ensure that the data of a DNS server is correct and the DNS server runs well. Availability issues make sure the DNS server is still available under any condition. Performance issues make sure the DNS server processes the requests more efficiently. Security issues deal with how to build robust servers to avoid problems such as illegal access and DDoS attacks. Different DNS server software or environment might lead to interoperability problems such as

IPv4/IPv6 protocols and between different versions of the DNS software programs (e.g., BIND v4/v8/v9). Finally, 512-byte limit issue exists for DNS query/answer UDP packets. Judging from this list of questions, we could find that some of the problem issues are related to general DNS servers, and others are related to IPv6 only (e.g., 512-byte limit, mail routing and application issues, etc.). To meet the requirements of TELECOM carrier level, an ENUM DNS needs more enhanced mechanisms on issues such as correctness, availability, performance, and security than IPv4 and IPv6 DNS.

3.5 ENUM DNS management issues

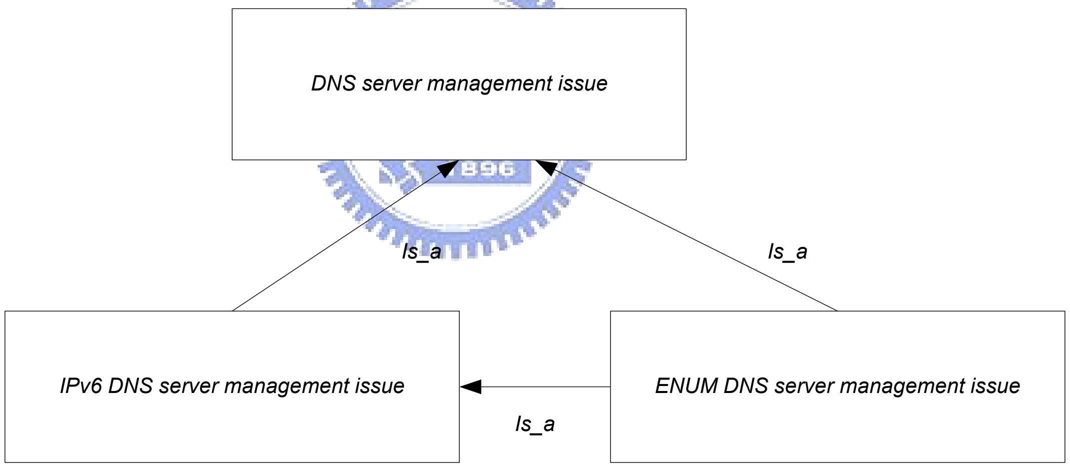

DNS server management issue
Is_a
Is_a
IPv6 DNS server management issue
ENUM DNS server management issue
Is_a

Fig. 3.1: The management hierarchy of general/IPv6/ENUM DNS server

Fig. 3.1 diagrams a simple hierarchy of management issues among general/IPv6/ENUM DNS servers. Just like the object-oriented language class hierarchy, the higher-level class is more general than the lower-level class. As a result, an IPv6 DNS server will inherit the management issues of a general DNS server. Similarly, the security measures in ENUM DNS servers should be much more reinforced than those in general DNS servers; that is, more resources (e.g., server hardware, bandwidth, man power, etc.) are supposed to be involved.

Traditional PSTN service should meet the TELECOM carrier level; i.e., high reliability, capacity and speech quality. Therefore, ENUM DNS servers should meet the above criterions as well, which differentiate them from ordinary DNS servers. Moreover, if an ENUM DNS is located on the IP network, most of the existing network attacks (e.g., DDoS attacks, system compromising, DNS spoofing, etc.) could possibly occur on the ENUM DNS. Since the DNS is the infrastructure of SIP/ENUM, if some ENUM DNS server fails, then the telephone number translation using ENUM DNS server will fail as well. Next, two scenarios will be given for illustrating the main ideas.

3.5.1 Scenario 1: Network attacks on ENUM DNS

In practice, DNS servers not only translate domain names into IP addresses, but also provide MX RR's for mail routing to deliver the mails. Moreover, on many Internet sites (e.g., SOHO people, etc.), all-in-one server (e.g., WWW, SMTP and DNS, etc.) is very common. However, the more unnecessary services are, the higher security threat is. For implementing ENUM DNS, it is supposed that the above situations should be avoided. For example, assume that a company X has its own ENUM DNS with all its subscribers' contacting information to provide the service. When someone needs to reach some subscribers of X, he/she queries the ENUM DNS to get the related contacting information. Suppose some attacker Y would like to shut down X's services by DDoS attacks. If there is not any protecting mechanism (e.g., DNS, router, etc.), Y might flood the ENUM DNS server with as many packets (e.g., mail, DNS, web, etc.) as possible and make it become un-available or the subscribers might wait for long time to get new connections.

Generally speaking, to secure the ENUM DNS, we need to take appropriate measures to implement and deploy the system architecture. First, it is necessary to separate other services from the ENUM DNS servers. Second, the ENUM DNS servers should be behind specific routers, separated from other internetworking equipments. Third, firewalls are required for helping filter out unwanted packets. Finally, network behavior analysis via IDS (Intrusion Detection System) for early detecting the anomalous traffic could help identify possible attack sources in advance.

3.5.2 Scenario 2: DNS spoofing

Assume that a commercial bank X has its own ENUM DNS and provides service phone numbers in its web pages, from which the customers could get expected service information. For example, suppose a phone number, +886-2-23456789, is put on some web page of X and the corresponding E.164 domain name is 9.8.7.6.5.4.3.2.2.6.8.8.e164.arpa. Basically, if some user Z from his/her ISP using DNS server Dz would like to call X's service, the X's ENUM DNS should map the domain name into the sip service, "sip:service@bankX.com.tw" and return it to Dz for Z's usage.

Now suppose there is no well protection mechanism on X's ENUM DNS, if a bad guy Y would try to get the customers' personal banking information of X by cheating, he might establish another faked site with similar web pages in advance and follow this by conducting DNS spoofing. For example, another different scenario, with DNS spoofing involved, about user Z using DNS server Dz might be as follows.

First, Y would set up his own ENUM DNS server containing some true authoritative data about Y and faked data about X. Second, Y might manage to bring the attention of Dz by using Dz directly (or indirectly via some other legal user of Dz, say, W) to query some domain data about Y. Third, the ENUM DNS of Y will return faked responses containing additional records of X (e.g., an NAPTR of 9.8.7.6.5.4.3.2.2.6.8.8.e164.arpa maps to sip:service@bankFake.com.tw., etc.) to Dz. Fourth, the poisoned data about X is put into the DNS cache of Dz. Finally, if someone (e.g., the user Z, etc.) using Dz would like to call X's service later, the phone call would be mis-directed and intercepted by Y.

Even though DNS spoofing problems had been identified, and some mechanisms had been proposed and implemented to address the problem on newer versions of DNS software programs; however, most DNS servers on many Internet sites only implement parts of these mechanisms, or even none at all, due to many problems such as performance and ignorance. Moreover, if we would like to adopt the ENUM DNS approach for providing commercial transactions in the future, it is also important to ensure the authenticity and integrity of the data by adopting DNS software with appropriate characteristics, which will be discussed later.

3.5.3 Scenario 3: Mailing errors due to the lack of reverse DNS entries

The mail server of a small company ***W*** worked fine for a long period time. However, due to the cost/performance considerations, the administrator was asked by the boss to move their Internet connection (e.g., originally with a leased line Internet connection) to another new ISP that provided cheaper ADSL links, with their mail domain name(s) kept unchanged. In the first few days, it seemed that all were OK. However, after that, users started complaining that they had mailing problems. While some users said that their outbound messages to specific destinations got bounced immediately each time, others complained that they got intermittent (e.g. sometimes successful, sometimes failed) bounced messages to many destinations. At first, the administrator suspected that the remote SMTP hosts might have some unusual (even unreasonable?) changes of the access control mechanisms against their mail host. However, after contacting many recognized administrators of some remote sites and having discussions with them, he finally got the solution to the problems.

Currently, there is a convention by many Internet sites to block SMTP connections from personal ADSL users since most of the SPAM messages were found to be injected from personal ADSL and dialup users [LT+03-1] [LT+03-2]. It turned out that their mail server, with a new ADSL link, had a reverse DNS mapping name under the ISP's ADSL-styled name. After changing it to another one different from the ADSL-styled format, the problem was fixed.

Chapter 4   DNS Knowledge Portal

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. In Chapter 4, we describe the main ideas (e.g., knowledge representation, etc.) on the design and implementation of the proposed DNS knowledge portal.

4.1 DNS knowledge Representation

As we may know, knowledge representation is one of the most central and familiar concepts in AI. Five distinct roles of knowledge representation are described in (DS+93). They are listed below:

- A knowledge representation (KR) is most fundamentally a surrogate.
- It is a set of ontological commitments.
- It is a fragmentary theory of intelligent reasoning.
- It is a medium for pragmatically efficient computation.
- It is a medium of human expression.

In our system, we adopt ontology representation and rules representation as the knowledge representation. From the above, we know that a knowledge representation is used as a substitution for the real world object. In principle, it is impossible for us to describe the real object completely because the one that could really denote the object is itself. In general, different knowledge representations focus on different views. Furthermore, different applications may need different representations on the same problem domain.

4.1.1 Ontology Knowledge Representation

An ontology is an explicit specification of a conceptualization [Gruber93]. 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 [CJ+99]. 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 [CJ+99] [GS93]. Furthermore, the graphical representation of ontology could simplify the communication between the domain experts and knowledge engineers.

As mentioned in [Fernandez99], 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 [FG+97], the authors of METHONTOLOGY explain that the life of an ontology moves on through the following states: specification, conceptualization, formalization, integration, implementation, and maintenance. Knowledge acquisition, 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 any state to other if some definition is missed or wrong. So, this life cycle permits the inclusion, removal or modification of definitions anytime of the ontology life cycle.

Tools are helpful to aid ontologists in constructing ontologies, and merging multiple ontologies since such conceptual models are often complex, multi-dimensional graphs that are difficult to manage. These tools also usually contain mechanisms for visualizing and checking the resulting models – over and above the logical means for checking the satisfiability of the specified models. ***Protégé-2000*** [NF+00] is an easy-to-use knowledge acquisition tool that could construct the domain ontology and achieve the interoperability with other knowledge-representation systems. In [CT+02-2], we built a DNS ontology using the ***METHONTOLOGY*** [FG+97] methodology and ***Protégé-2000*** [Gennari+03] system from scratch. The knowledge model of ***Protégé-2000*** is frame-based and the ontology built consists of classes, slots, facets, instances. The class elements are used to describe the concepts, from which we could build the class hierarchy of the taxonomy. For example, Figure 4.1 shows a diagram about the DNS class mentioned above. In the DNS ontology, since both master and slave DNS servers are DNS servers, they both belong to the subclasses of the DNS authoritative server class and thus inherit the DNS property.

Slots in Protégé-2000 describe the properties of classes and instances, such as the configuration of the DNS server, or the software version of the DNS server program. A slot could be created without being attached to a specific class. For example, a version slot could be used to denote the version of the ISC BIND or the Microsoft DNS server software. On the other hand, when we need to bind one slot to a specific class, it could have some value. For example, if we attached the version to the BIND software, it could have some value of 8.2.2., 9.2.1, or other similar one.

Facets in Protégé-2000 are used to define the constraints of the slots. For example, the cardinality of the version attribute in the DNS ontology is single numeric value and its type is symbol. We also could define the minimum and maximum value for the numeric slots. In this way, we could set up the constraints of cardinality or the value type of the specific slot. In addition to the ontology classes, slots and facets, the physical elements of the ontology are instances. In other words, the ontology classes, slots and facets define the skeleton and instances element fill in the physical information. In essence, when we would like to design database, we would first define database schema. The ontology class information is similar to the database schema. For example, as shown in Fig. 4.2, the employee and department tables define the employee table attributes and department attributes. In addition, the foreign key information associate employee table with department table. After the database schema design, we may insert the real data into the tables. For example, we may insert a data record with the following information:

EmployeeID: 1234

EmployeeName: Alice

Age: 29

Salary: 50,000

DepartmentID: 1

DepartmentID: 1

DepartmentName: RD

The data record above represents the instance of the database schema. We could manipulate the data by SQL command (e.g. SELECT, UPDATE, or DELETE etc.). For example, if we would like to retrieve the employees whose salaries are more than 40,000, we could use the following SQL command:

```
SELECT * FROM EMPLOYEE WHERE SALARY > 40000
```

Protégé provides the similar query mechanism for knowledge retrieval. In essence, the ontology class information and instances are similar to the database schema and data records respectively. In addition to the retrieval functionality, the ontology KBS provides the logic reasoning mechanism. For example, as shown in Fig. 4.3, the animal ontology hierarchy shows the hierarchy information. “Mammal” is a kind of “Animal” and “Person” is a kind of “Mammal”. After the logic reasoning process, we could infer that “Person” is a kind of “Animal”.

In essence, database ER model diagram could give us the overview of the application domain. Ontology could play the same role during the knowledge construction process. Ontology could be used as the communication media between domain experts and knowledge engineers. In addition, the graphic representation of ontology is more user friendly representation and that could improve the knowledge acquisition. Therefore, in our design, ontology often plays an important role during knowledge acquisition and system construction. In general, if constructing ontology KBS, we would need the instances to fill in the KBS and based on the ontology information for reasoning. However, in some application domain, the instances do not exist. In DNS domain, although we could define the DNS ontology properties and relationships, the instances of DNS ontology are meaningless. For example, our DNS ontology defines a DNS class which consists of NS record property, domain name property and MX record property. The instance would be:

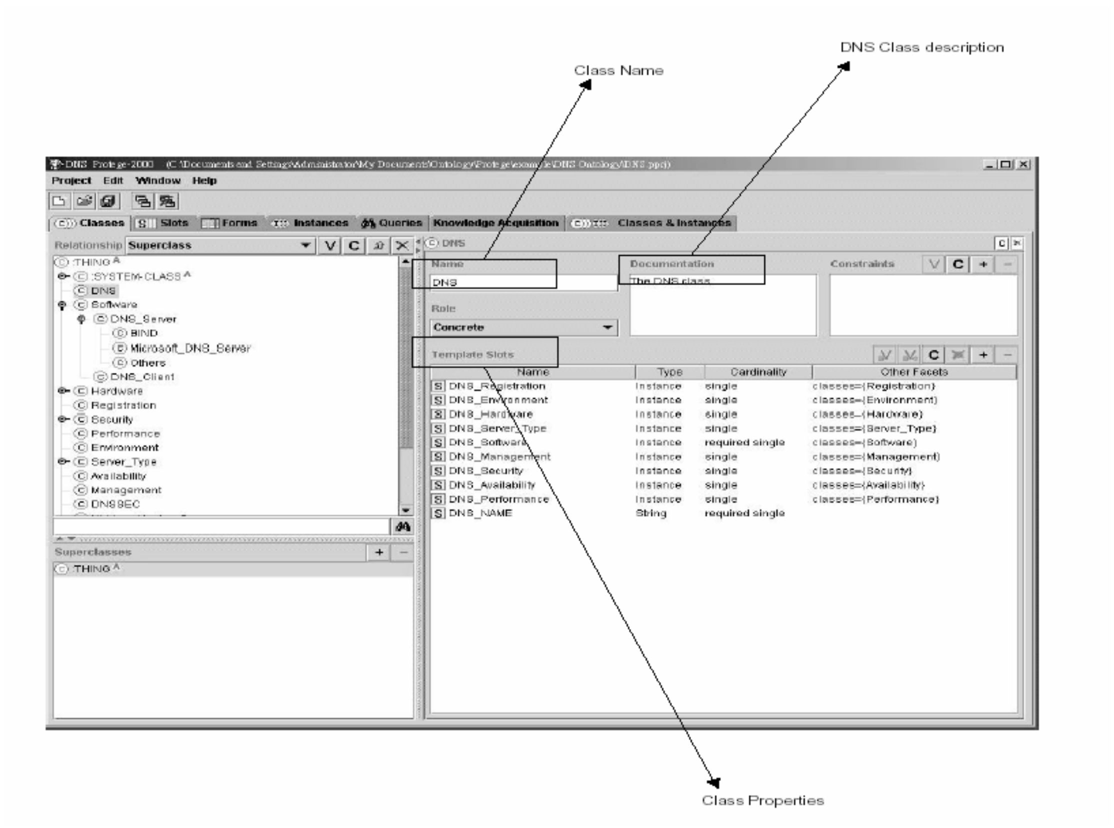
DNS

Domain_Name: (the domain name of DNS server)

NS: (the NS record information of DNS server)

MX: (the MX record information of DNS server)

However, the above information would exist when diagnosis system retrieves by querying users' DNS server. In other words, it is meaningless to store arbitrary DNS server information. What we are interested is to infer the diagnosis result based on the above value. For example, if the number of users' DNS NS records is less than two, we could infer that SPOF problem exists in users' DNS. Therefore, we propose a hybrid knowledge model for DNS domain, which considers ontology knowledge and rules knowledge.



Name: DNS

Documentation: The DNS class

Constraints

Role: Concrete

Template Slots

Name	Type	Cardinality	Other Facets
DNS_Registration	Instance	single	classes={Registration}
DNS_Environment	Instance	single	classes={Environment}
DNS_Hardware	Instance	single	classes={Hardware}
DNS_Server_Type	Instance	single	classes={Server_Type}
DNS_Software	Instance	required single	classes={Software}
DNS_Management	Instance	single	classes={Management}
DNS_Security	Instance	single	classes={Security}
DNS_Availability	Instance	single	classes={Availability}
DNS_Performance	Instance	single	classes={Performance}
DNS_NAME	String	required single	

Fig. 4.1: Building the DNS ontology using Protégé-2000

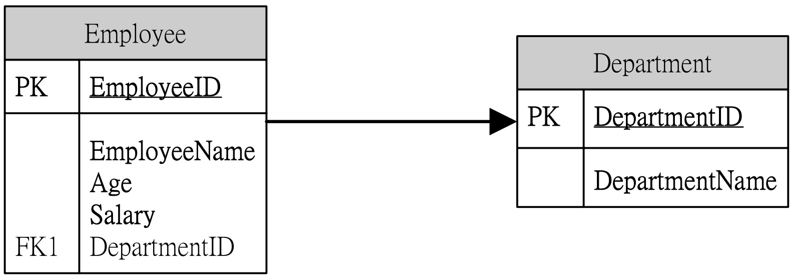


Fig. 4.2: Employee and department table schema

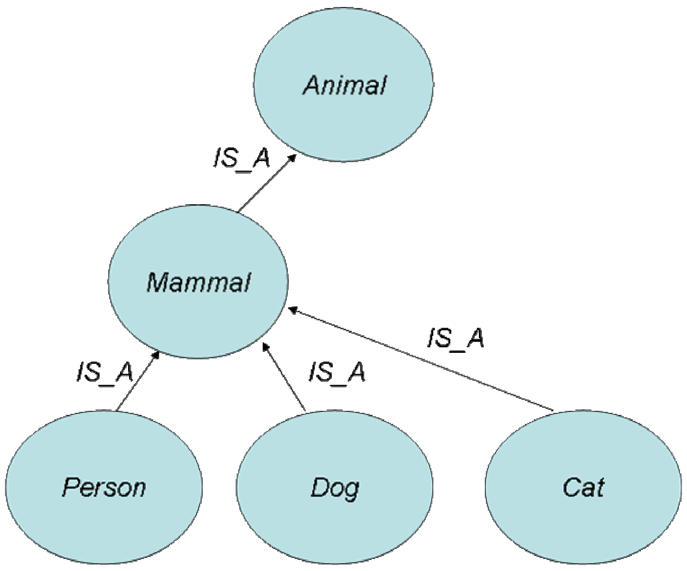


Fig. 4.3: Animal ontology hierarchy

4.1.2 Rule-based Knowledge Representation

One of the most popular approaches to knowledge representation is to use production rules, sometimes called IF-THEN rules. The basic form of the rule representation is:

If <condition> Then <Action>

When the incoming faces meet the condition, the inference engine would infer that the rule should be fired and the action part would be active. Some benefits of the IF-THEN rule representation are that they are modular, each defining a relatively small and, at least in principle, independent piece of knowledge. In addition, the IF-THEN is similar to natural language and it is easily understood. Furthermore, the

IF-THEN rules are powerful to define the mechanism for the application domain. For example, most of the firewall software is typical rule based system. The network administrator defines the firewall rules to filter out unwanted network packets or protocol. Due to the network attacks, most of the network administrators would only allow web access and the pseudo rule may be:

If *the port of destination server <> 80* Then *Reject*

In addition, many network services (e.g. IDS (Intrusion Detection System), antiSPAM software etc.) adopt rules as the engine to perform the jobs. Furthermore, rule representation is suitable for DNS domain as well. In DNS diagnosis system, we would like to diagnose DNS problems from user' DNS configuration. The DNS configuration information could be viewed as the facts and our system would start the diagnosis process. In essence, the whole process is a typical forward reasoning process. For example, we could define the SPOF (Single Point Of Failure) rule as follows:

If number of *NS* record < 2 Then SPOF

The fact section of the above rule is the number of *NS* record, which could be retrieved from users' DNS configuration. The rule representation is more readable for DNS administrator. Hence, rule representation is suitable for DNS diagnosis system. However, the rules extraction is not easy and rule management is difficult when the number of rules become huge. Therefore, the mechanism for rules extraction and management is required.

4.1.3 Hybrid Knowledge Model

Database schema design is an important process when we would like to construct a system. In general, the logics of the database application are often related to the database design. The database schema reflects the attributes required during the system process. In essence, the graphical representation of database ER model could be used as the communication media between the DBAs and the software engineers. In essence, knowledge acquisition is often the bottleneck of KBS. It is not easy to extract knowledge directly from domain experts. Therefore, some mechanism is required during the knowledge acquisition. In essence, ontology representation is easily understood by domain experts and knowledge engineers. The concept hierarchy, concept attributes and relationships are similar to the object-oriented design or database schema design. In addition, many existing ontology tools (such as Protégé) can simplify ontology construction. Therefore, just like the role of database schema in software engineering, ontology representation is also suitable for knowledge engineers and domain experts to model the domain knowledge.

As described above, rules representation is more suitable for DNS domain but the rule extraction is not a straightforward process. In [LT+04-1], we proposed an ontology-driven model for rule extraction. The whole process is to facilitate the domain experts to extract the rules by the help of ontology. The ontology could guide the rules extraction and simplify the whole process. In addition, ontology hierarchy information could represent the problem decomposition process. For example, in SPOF problems, we could further decompose SPOF into single server problem and single network problem. Model-tracing tutoring [AB+00], which is based on the ACT [AC93] theory of skill acquisition, makes use of production rules to simulate the skills. In practice, model-tracing tutoring has been applied in many domains (e.g. LISP, algebra, etc.). However, the construction of model-tracing tutoring is not straightforward as well. Therefore, in [LT+04-3], we propose an ontology-based DNS model-tracing tutoring model which helps knowledge engineers to construct the skeleton of the model-tracing tutoring and extract the production rules to simulate users' behavior and skills. In DNS knowledge portal, we adopt both ontology and rule knowledge representations to model the knowledge. The advantages of the hybrid knowledge model are as follows.

- Ontology representation could make domain problem modeling more easily.
- Ontology could facilitate the KBS rules extraction and model-tracing tutoring production rules extraction.
- DNS diagnosis could be addressed by rules
- Ontology could help the knowledge engineers construct the skeleton of model-tracing tutoring

1896

4.2 Ontology-based Learning Sequence Construction

As described above, ontology could represent the knowledge structure. In addition, the learning sequence of the tutoring system often could reflect the course structure. For example, in algebra domain, the symbolization course should be introduced before the algebra equation course, since the basic element of the algebra equation is the symbolization of the unknown element. In general, the structure of the course needs many domain experts involved. In addition, to provide the students the course content adaptively, individualized learning is important as well. Therefore, the mechanism which could help the domain experts during the course structure construction process is required. As described above, the ontology structure could simplify the communication between domain experts and knowledge engineers. In addition to ontology knowledge, rules representation is appropriate in DNS domain. In general, the examples are very important for DNS administrators. Because the examples could provide concrete DNS configuration, the DNS administrators could apply the examples on their own DNS configuration with some little modifications. In addition to the examples, the quiz could help the domain experts to verify whether they understand the course or not. In this section, we would describe how to apply ontology and meta-rules to build the learning sequence scheme.

As shown in Fig. 4.4, the whole process consists of ontology-based learning sequence construction model, meta-knowledge extraction module and example and quiz annotation module. As shown in Fig. 4.5, the ontology-based learning sequence is used to generate basic DNS course scheme. In essence, to meet the requirement of individualized learning, the DNS course scheme should be adaptively presented based on different criteria (e.g., students' profile, students' behavior, students' background knowledge, etc.). Different domain may need different criteria, so we focus on the DNS ontology in this section. Algorithm 4.1 shows that the input is the domain ontology and the output is the basic course scheme.

Algorithm 4.1: Basic learning sequence construction algorithm

Input: The domain ontology

Output: The basic course scheme

Step 1: Take the core class as the *now-class*.

Step 2: Find available *relationship and associated-class pairs* of the now-class.

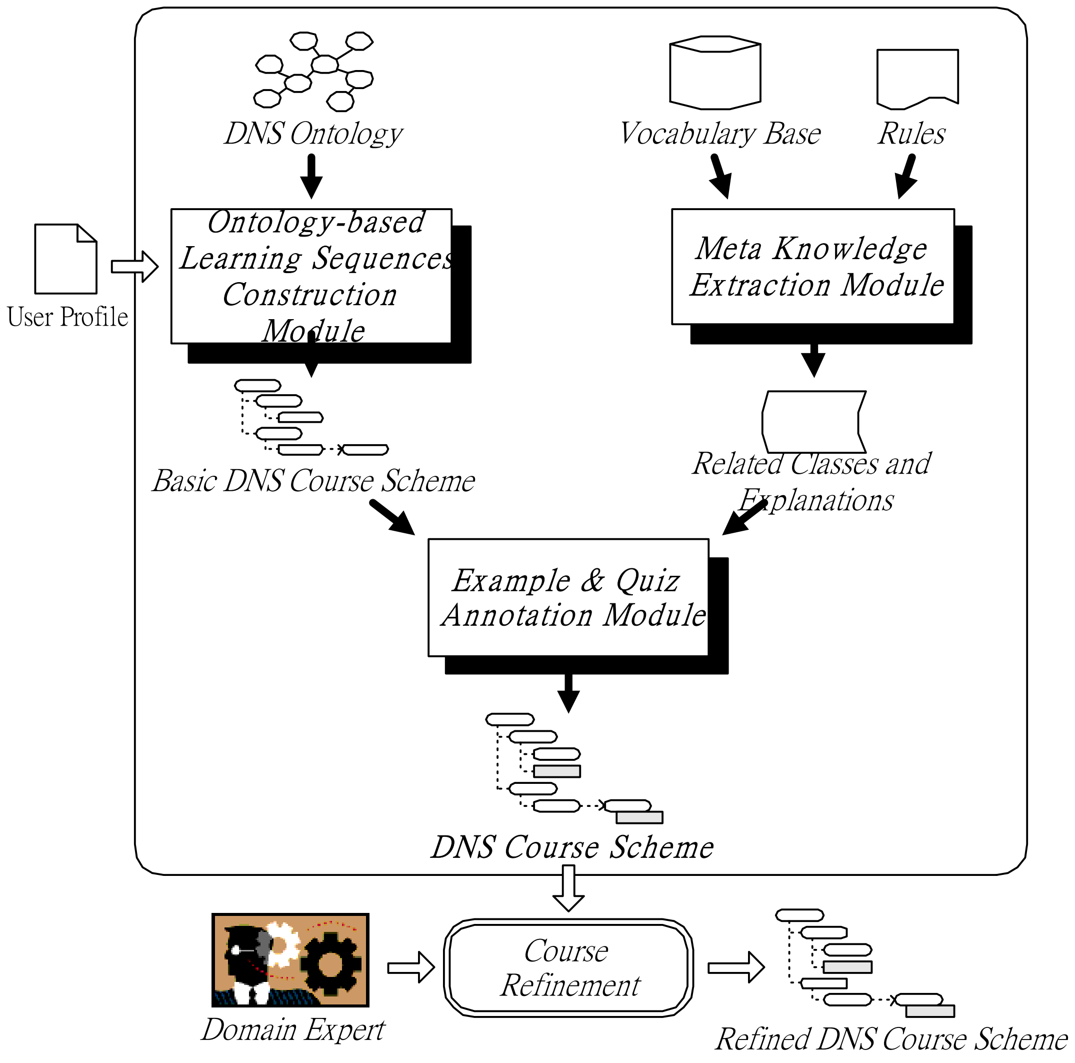
Step 2.1: Find all the relationship and associated-class pairs of the now-class.

Step 2.2: If the relationship is not available, then eliminate the relationship and associated-class pair.

Step 3: Sort the relationship and associated-class pairs by the priority of the relationships.

Step 4: According to the order of the sorted list, construct the corresponding learning sequences.

Step 5: Take the associated-class as the now-class and go to Step 2 in turn.



DNS Ontology
Vocabulary Base
Rules
User Profile
Ontology-based Learning Sequences Construction Module
Meta Knowledge Extraction Module
Basic DNS Course Scheme
Related Classes and Explanations
Example & Quiz Annotation Module
DNS Course Scheme
Domain Expert
Course Refinement
Refined DNS Course Scheme

Fig. 4.4: Ontology-based learning sequence construction

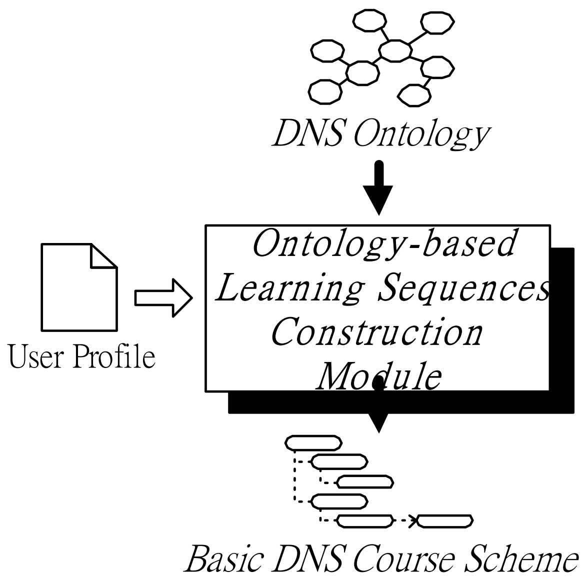


Fig. 4.5: Ontology-based learning sequence construction module

In addition to the ontology representation, the rules representation could provide us some information about the quiz or example construction. In the example annotation section, the examples are attached on rules. For example, the DNS SPOF rule is listed as follows:

IF *number of NS records* < 2 THEN *SPOF* = true

Explanation = "DNS availability"

Based on the above rule information, the domain experts could provide the related examples and explanation which could be attached on the rules. As shown in Fig. 4.6, the SPOF rule is related with NS records and the SPOF result and the NS record is the fact section which could be viewed as the reason of SPOF. In general, when users learn NS record course, they still do have the knowledge about SPOF problem, so that it would be better if the example of SPOF is presented after SPOF course is introduced. Furthermore, NS record and SPOF are both DNS ontology concepts and they could be located from the DNS ontology vocabulary. As shown in Fig. 4.7, the

DNS ontology vocabulary and rules are the inputs and the meta knowledge extraction module would extract related classes and explanations.

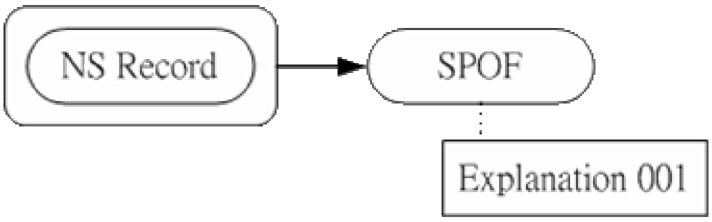


Fig. 4.6: Example annotation in SPOF rule

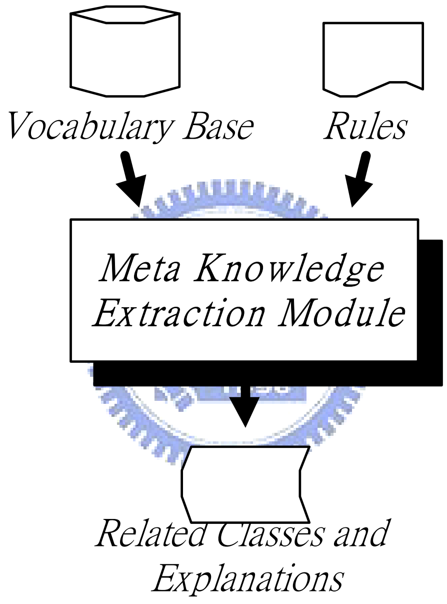

Vocabulary Base
Rules
Meta Knowledge Extraction Module
Related Classes and Explanations

Fig. 4.7 Meta knowledge extraction module

After the DNS basic course skeleton, related class and explanation are discovered, we would start the process of annotation. Algorithm 4.2 shows example annotation algorithm. The example annotation algorithm would traverse the DNS ontology tree to discover the appropriate node related to the explanations. In other words, after example annotation process, the examples would be located on appropriate course. As shown in Fig. 4.8, the annotated result would be verified by domain experts and the domain experts would refine the course scheme if needed.

Algorithm 4.2: The example annotation algorithm

Input: The basic DNS course, related classes and explanations.

Output: The DNS course with example annotated.

Step 1: Start from the beginning class of the basic DNS course. Take this class as the now-class.

Step 2: Check each rule, and mark the related class which is the same as the now-class.

Step 3: If all the related classes of a rule have been marked, then annotate the explanation as an example to the now-class.

Step 4: Go through the learning sequences, take the next class as the now-class, and go to Step 2.

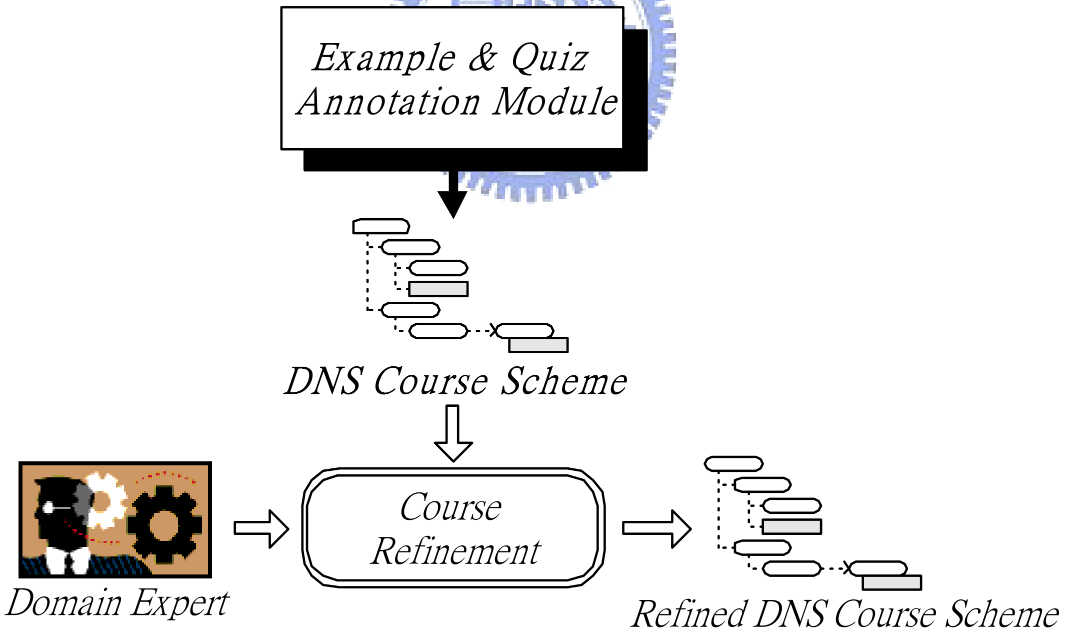


Fig. 4.8: Example and quiz annotation module

4.3 Diagnosis-Learning-Search Model

We have started to provide DNS diagnosis service since 2003. The diagnosis service could help the DNS administrators diagnose their DNS servers and most of the feedbacks are positive. However, many users feel that more instructions are required after the diagnosis result presented. In general, the target users of DNS diagnosis system are the DNS administrators who have built DNS servers. Furthermore, since DNS is the infrastructure of Internet, many Internet services rely on DNS (e.g. WWW, email etc.). Therefore, if they meet the DNS configuration problems, they would need the solutions as soon as possible. In essence, DNS diagnosis system would fulfill their requirements. In other words, DNS is a problem-driven domain and the combination of diagnosis system and tutoring system is required for some users who would like to know the DNS operational model in more detail.

As described above, model-tracking tutoring focuses on the problems issues as well. In addition, DNS configuration could be viewed as users' behavior and the diagnosis system could retrieve the configuration information through network DNS query. Therefore, the DNS diagnosis system could act as the quiz of the DNS and the DNS configuration information is users' answer. As shown in Fig. 4.9, the whole diagnosis-tutoring model could be summarized as following:

1. Users start the DNS diagnosis service.
2. If the users are interested in more information about the operational model, they could start the tutoring service. The tutoring service would adopt users' configuration information, which is retrieved from DNS diagnosis service, as users' behavior.
3. User could start to navigate the tutoring materials.

4. The users could modify their own DNS configuration and start the diagnosis to test whether they understand the operational principles.

In addition to diagnosis-tutoring model, search is another service of existing DNS knowledge portal. The search service could search articles in the file system, data records in the database, or other information sources. Therefore, when the users would like to find out required information, search service would facilitate a lot. Most of the traditional search system is based on keyword search without semantics embedded in the search string and that may lead to inappropriate result. In theory, ontology could represent the semantics of the terms. Therefore, we adopt the ontology as the semantics resolution mechanism to improve the search capability. As shown in Fig. 4.10, the whole process is as follows:

1. Users submit the query string to the search service.
2. The search service starts to inference the query string based on DNS ontology and starts to search the data source based on the inference result.
3. The search service returns the search result to the users.

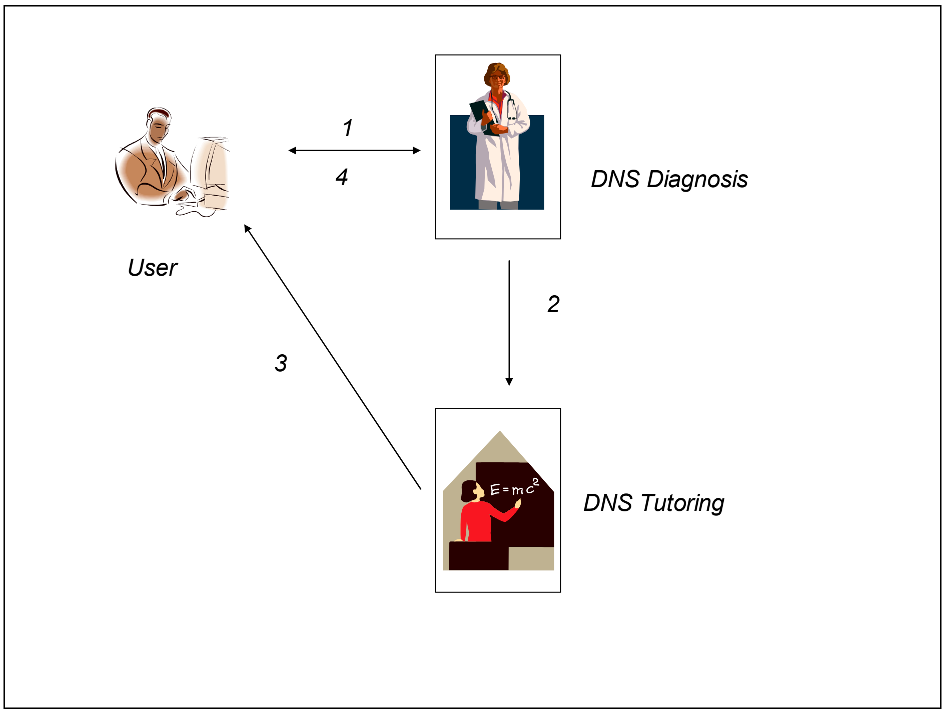


Fig. 4.9: Diagnosis-Tutoring model

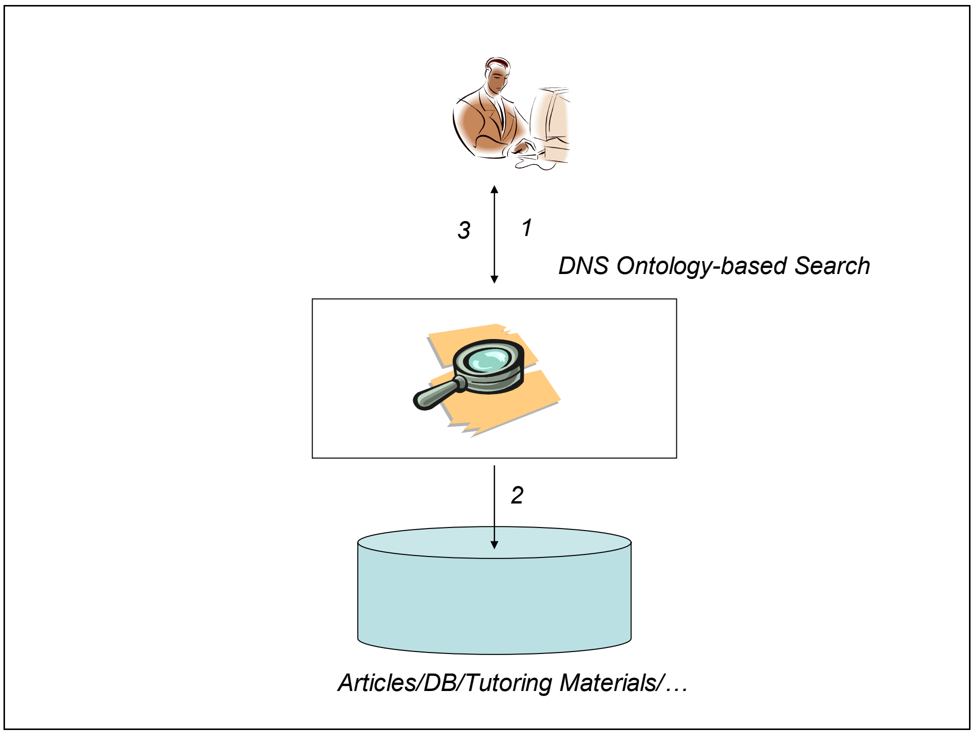


Fig. 4.10: Ontology-based search service

Chapter 5　DNS Ontology and Ontology-Driven Model

As mentioned in Section 4.1, ontologies are becoming an important mechanism to build knowledge-based information systems. In essence, ontology representation is suitable for communications and natural for human thinking. The role of ontologies is to capture domain knowledge and provide a commonly agreed upon understanding of a domain. In this chapter, we would focus on describing the ontology-driven model for diagnosis rules extraction, model-tracing tutoring, SCORM learning sequence construction and ontology-based search respectively.

5.1　ENUM DNS Knowledge and Ontology

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 [CJ+99]. The role of ontologies is to capture domain knowledge and provide a commonly agreed upon understanding of a domain. The common vocabulary of an ontology, defining the meaning of terms and their relations, is usually organized in a taxonomy and contains modeling primitives such as concepts, relations, and axioms [HS+97].

In general, Ontology modeling is similar to object-oriented design modeling. In principle, we could view all the entities in the world as objects or concepts. When we would like to describe the objects or concepts, we could describe their attributes or slots. In addition to the internal attributes, we could represent the interaction between the objects by using relationships mechanism.

In Chen et al. (2003), we built a DNS ontology, which was used to fulfill the skeleton of our KBS. The domain knowledge of our KBS has been described through a semantic network as shown in Fig 5-1. The taxonomy of DNS concepts could help us classify DNS and related knowledge.

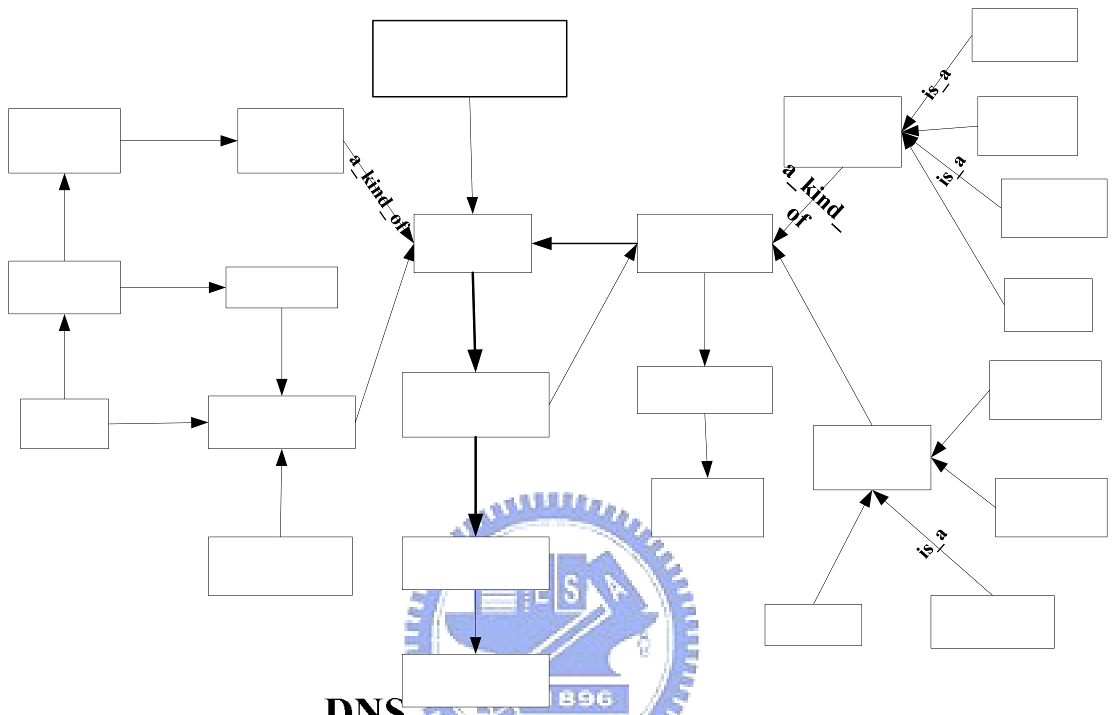


Fig. 5.1: Domain knowledge in the KBS to aid the DNS management

As described above, we could view the information from both the server and client sides. From the former, DNS is the main concept, which has many services (e.g., DNS registration, DNS query resolving, etc.). And, we could further divide the services into central-service and non-central-service. From the latter, what the client does is to send queries to the DNS server and we could find that there are many common query types such as A, MX, and PTR. NAPTR [HS+99] is a new query type, so NAPTR is located in specialty concept. In addition, the *resource* concept describes the required resources for DNS server.

In the following sections, we would focus on describing the ontology-driven model for diagnosis rules extraction, model-tracing tutoring, SCORM learning sequence construction and ontology-based search respectively. In addition, we would like to integrate all the services into the portal system. It is supposed that the integrated services could help DNS administrators to solve DNS problems and learn DNS related knowledge more efficiently.

5.2 Ontology-driven model for rule extraction

For dealing with maintenance issues, knowledge classes 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.

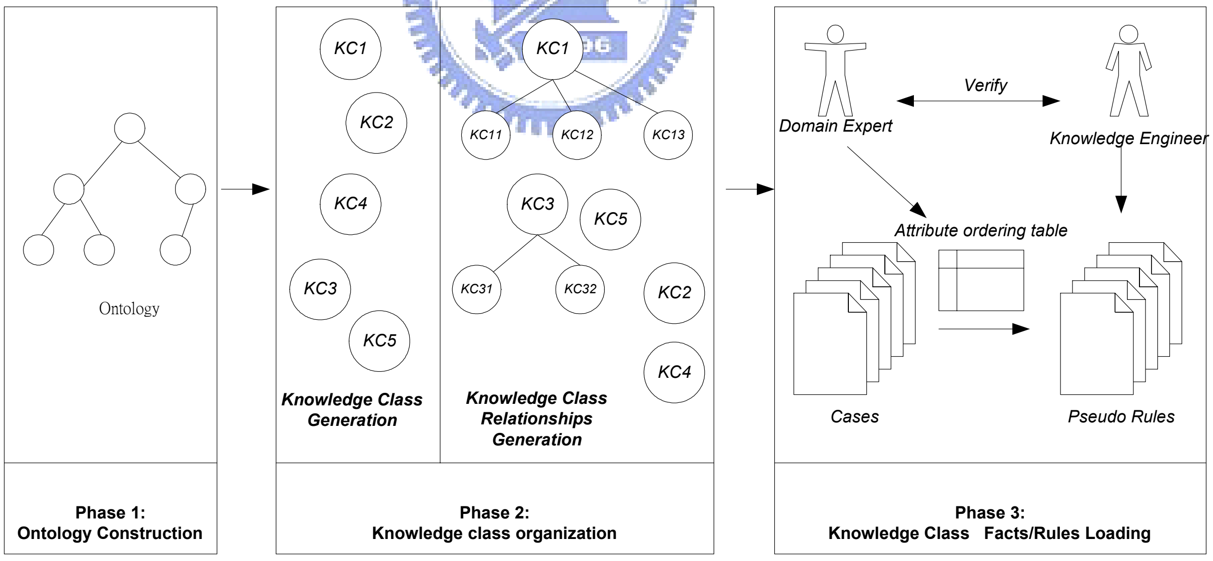

Ontology
KC1
KC2
KC4
KC3
KC5
Knowledge Class Generation
KC1
KC11
KC12
KC13
KC3
KC5
KC31
KC32
KC2
KC4
Knowledge Class Relationships Generation
Domain Expert
Verify
Knowledge Engineer
Attribute ordering table
Cases
Pseudo Rules
Phase 1: Ontology Construction
Phase 2: Knowledge class organization
Phase 3: Knowledge Class Facts/Rules Loading

Fig. 5.2: Ontology to DRAMA knowledge class

As shown in Fig. 5.2, we propose an ontology-driven model for rules extraction. The whole process is described as follows:

- **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 [HS+97]. 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 [Cockburn97]. Fig. 2.3 shows a snapshot of the DNS ontology. The DNS construction algorithm is summarized as follows:

Algorithm 5.1: DNS ontology constructing algorithm

Input: Every kind of DNS cases.

Output: DNS Ontology.

Step1: Build the Skeleton DNS ontology (top-down)

Step2: Initiate (or conduct) use case modeling

Step3: Conduct the attributes and relation extraction.

Step4: Merge the ontological components collected in Step1 and Step3 above.

Step5: Experts verify the ontology.

Step6: After experts' verification, the DNS ontology is constructed to cover DNS domain knowledge.

- **Knowledge class organization phase**

As described above, since the knowledge class of NORM knowledge model is based on the concepts, the transformation between the ontology concept class and the knowledge class 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 knowledge classes as described in Fig. 5.3.

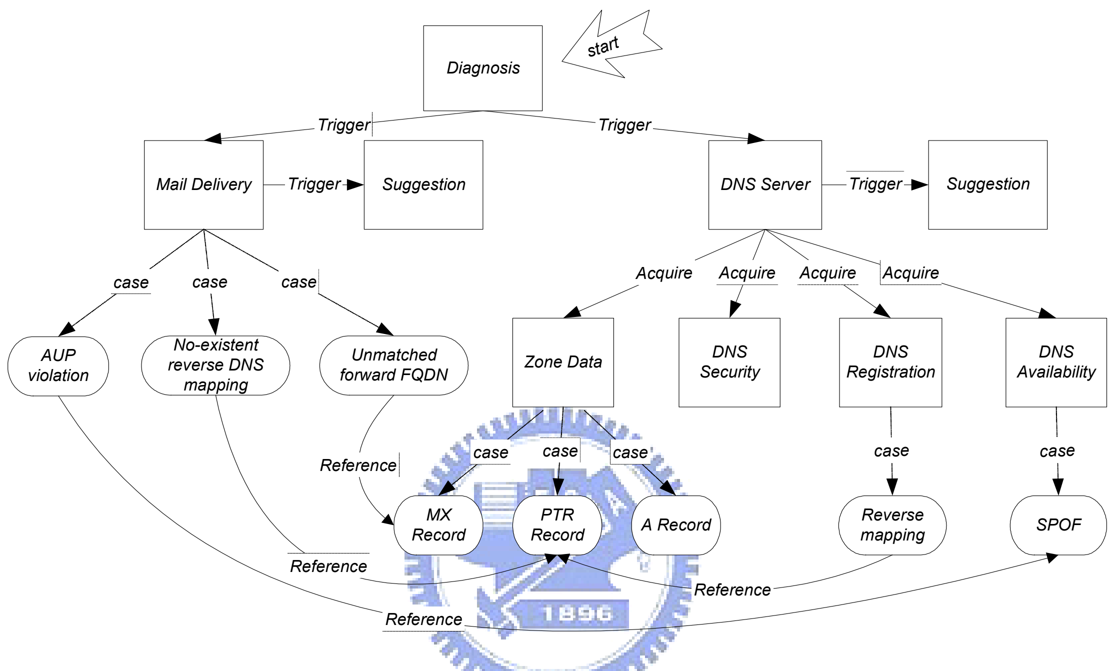


Fig. 5.3: The knowledge class structure of diagnosis service

In the process of DNS construction, we should consider DNS issues including availability, performance and registration, etc. Fig. 5.3 shows the inference scheme of diagnostic examples about DNS-related mailing problems. The rectangles mean KC'es in NORM and the rounded rectangles mean cases of some particular KC'es. In addition, the solid lines indicate relations of the KC'es and their correlated cases.

As specified in Fig. 2.3, 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 thesis, 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 knowledge classes 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:

Algorithm 5.2: Ontology to knowledge class transformation algorithm

Input: DNS ontology

Output: DNS Knowledge Classes and the relationships of Knowledge classes

Step1: Transfer the needed ontology concepts into knowledge classes: For each DNS ontology concept, we could transfer the concept into the knowledge class.

Step 2: Define or identify the relationships between the knowledge classes.

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 knowledge classes ***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 knowledge classes ***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 knowledge classes ***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.

■ **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 [GS92], 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 thesis, we make use of repertory grid like concept to help elicit knowledge. Table 5.1 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 5.2 shows the ordering table of single server and single network.

Table 5.1: SPOF case description

Description	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.		
Case NO.	Case Name	Description	Actor
Case 1	Single DNS Server	You have only one DNS server listed for your domain	DNS
Case 2	Improper DNS configuration	Of the servers listed for your domain, only one of them is properly configured for your domain.	DNS
Case 3	The same physical position	All of the DNS servers that are both listed in your domain registration and properly configured for your domain reside on the same physical subnet, or in the same physical location, or otherwise rely on any one single piece of equipment.	DNS
Case 4	The same router	All the DNS servers are behind the same router	DNS

Table 5.2: Attribute ordering table for single server/single server cases

	Single Server	Single Network
NS Record	5	1
MX Record	1	1
A Record	1	1
PTR Record	1	1
SOA Record	1	1
Physical Location	1	5
CNAME	1	1
Zone Data	1	1

Table 5.3: Attributes and values of NS Records for Single Server

Attribute	Value
The Number of NS Record	< 2

The IP address of NS Records	Master DNS and Slave DNS are not alive.

Table 5.4: Attributes and values of physical location for Single Network

Attribute	Value
Master DNS Server Location and Slave DNS Server Location	In the same network location
Master DNS Server Location and Slave DNS Server Location	Behind the same router

Table 5.5: SPOF pseudo rules

Case Name	Rule
Single DNS Server	If number of NS Record <2, Then SPOF (Single Point Of Failure)
Improper DNS configuration	If Master DNS and Slave DNS are not live, Then SPOF
The same physical position	If master DNS Server and slave DNS server are in the same network location, Then SPOF
The same router	If the location of Master and Slave DNS servers are behind the same router, Then SPOF

■ **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. Table 5.3 and Table 5.4 show the attribute/value pair tables for single network and single server respectively. Finally, KEs could generate the pseudo rules, as shown in Table 5.5, 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.

Algorithm 5.3: 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.

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.

5.3 DNS Ontology-based Model-Tracing Tutoring

In general, ontology representation is appropriate for knowledge modeling. For example, the DNS problem taxonomy structure could provide the DNS problem space. However, as we know, the rule-based representation is more appropriate when the problem domain can be described clearly and well modeled. In essence, DNS diagnosis system is triggered by rules and users' DNS configuration is acted as the facts of rules. Users input the DNS configuration data when constructing DNS servers, so the DNS configuration information could reflect users' activities. For example, when the diagnosis system finds the fact, only one NS resource record listed in the user's specified DNS configuration zone, it will fire the single-server rule under the SPOF knowledge class (i.e., the single-server rule firing could infer that the SPOF problem exists). Therefore, ontology hierarchy information could provide the possible problem-solving space and the rules could be used to model users' activities.

In essence, DNS problem domain is very complex and varies greatly on different sites because too many things, like management strategies and resources, need considering [Bellovin95] [Bc+01] [CERT00] [CJ+99] [DNSBL03] [Faltstrom03] [Kumar+93] and most DNS administrators are primarily interested in the issues related to their DNS problems. Hence, DNS could basically be classified as a problem-driven domain. On the other hand, model-tracing tutoring tries to model users' behaviors by production rules and focus on tracing the problem issues as well. Therefore, model-tracing tutoring is very suitable to apply on the DNS domain because of its problem-driven characteristics.

In general, the construction of model-tracing tutoring needs domain experts to help analyze the domain problems and decompose the problems into sub-problems to simulate users' activities during the problem-solving process. Usually, this is not a straightforward job. Therefore, as shown in Fig. 5.4, we propose an ontology-based model-tracing tutoring structure construction model for facilitating the model-tracing tutoring and the whole process is described as follows:..

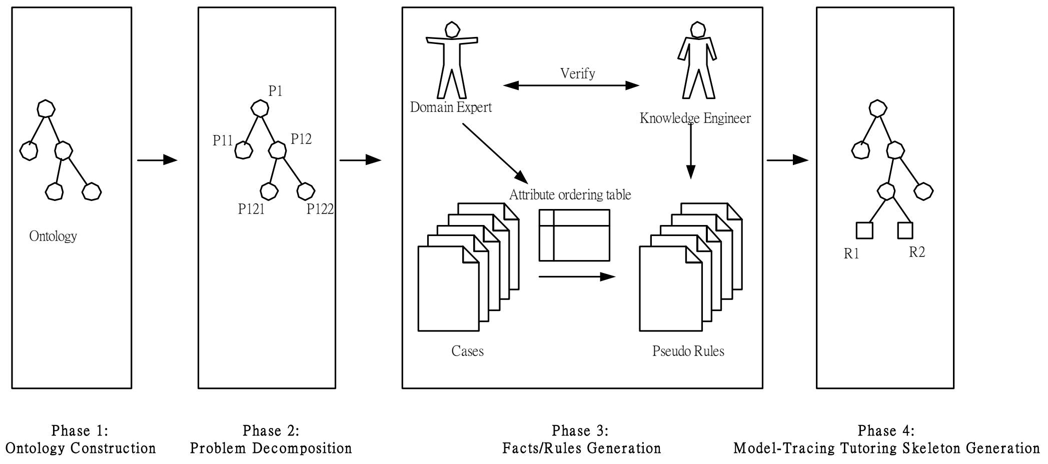


Fig. 5.4: Ontology-based model-tracing tutoring structure generation

(i) Ontology Construction Phase

As described in Section 5.2 Ontology Construction Phase.

(ii) Problem Decomposition Phase

As described above, model-tracing tutoring decomposes the problems into sub-problems and tracks students' progress and keeps them within a specified tolerance of an acceptable solution path. Therefore, if the focus of ontology is on the application problem issues, the problem decomposition process could be facilitated from ontology hierarchy information. For example, in DNS problem ontology, "Is-a" relationship exists between single-server concept and SPOF concept. That is, we could say that a single-server concept is a specialization case for SPOF and that could be further inferred that single-server problem is a sub-problem of the DNS SPOF problem. Hence, if we focus on SPOF problem, we could decompose SPOF problem into "Single Network" and "Single Server" sub-problems.

(iii) Facts/Rules Generation Phase

As described in Section 5.2 Facts/Rules Generation Phase

.

(iv) Model-Tracing Tutoring Skeleton Generation

As described above, ontology hierarchy information could be used to construct the skeleton of model tracing tutoring. Furthermore, the fact section of rules could reflect users' DNS configuration activities information. In addition, we need to interview domain expert for the correct configuration for each problem. Finally, we could generate model tracing tutoring for SPOF problem as shown in Fig. 5.5. Fig. 5.5 shows the DNS diagnosis knowledge class structure for SPOF knowledge class and model-tracing tutoring production rules structure for SPOF problem. For example, when the users fire rule R1, that means SPOF problem happened in users' DNS server and the result could infer that the users may not have knowledge about DNS SPOF problem. Furthermore, it also gives us the clue for providing appropriate teaching material or online help.

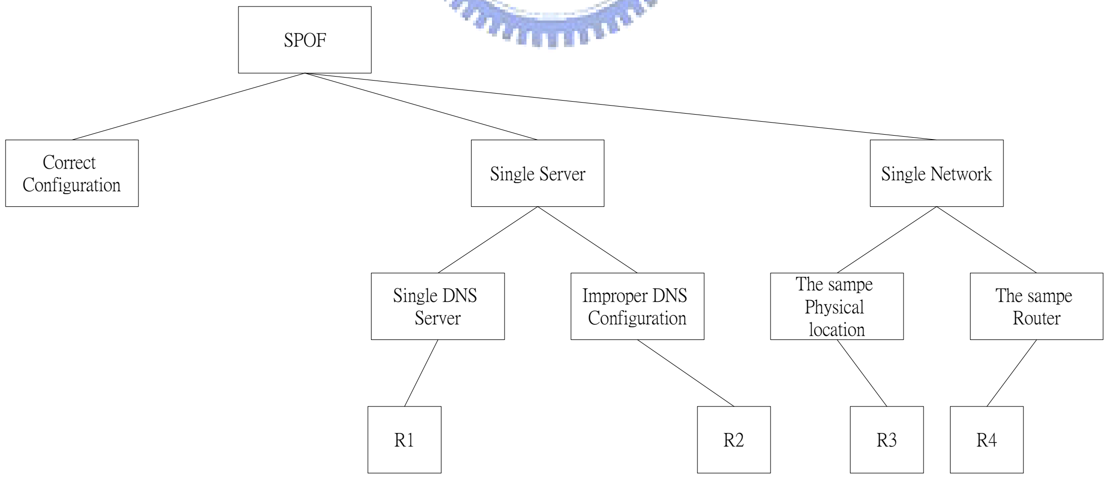


Fig. 5.5: Model-tracing tutoring for SPOF problem

5.4 DNS Ontology-based Search System

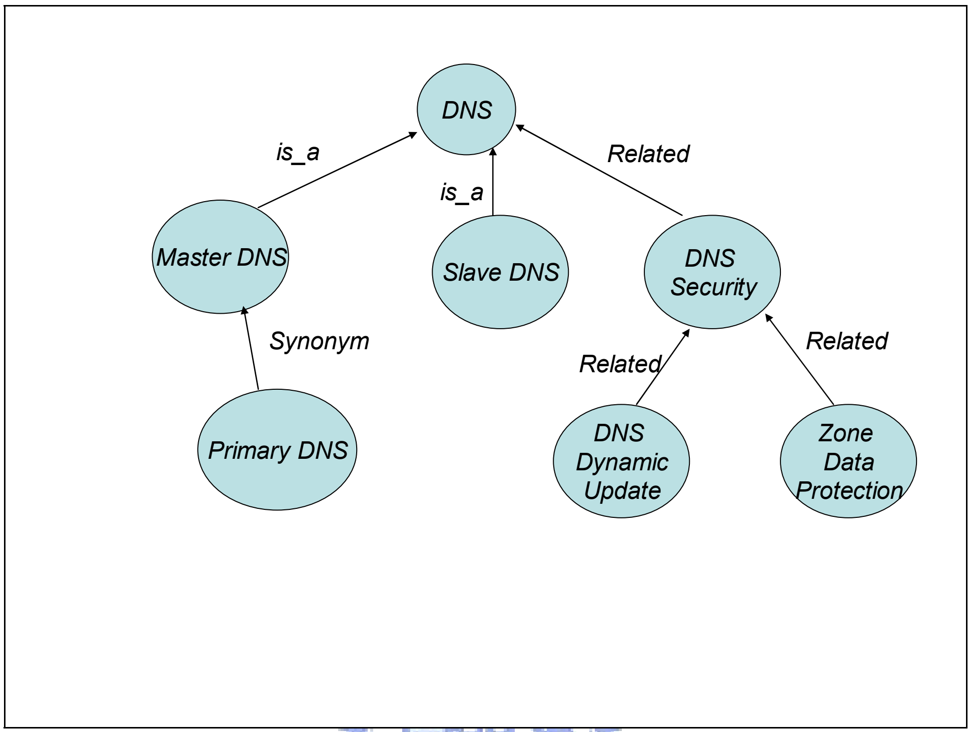


Fig. 5.6: DNS ontology examples

Most traditional search systems compute the similarities between objects (or concepts) based on the term frequency (TF) or inverse document frequency (IDF). However if we consider only the term, we would miss the semantic information of the term. It is not easy to take into account the term semantics information directly. Especially when we are not familiar with the domains, it is difficult for us to describe the terms correctly. For example, many issues (e.g. DNS spoofing, DNS zone data protection etc.) exist under DNS security issue. However, for most of the users, what they could describe is the term "DNS security". In other words, the general terms expression is easy for most of the users. Furthermore, the semantics information is important as well. For example, there maybe exist synonyms for every domain. However, if we consider only keyword mapping, the synonyms of the query string will be ignored and that may lead to information loss. Therefore, a system which could expand users' query string based on the background knowledge and understand the term semantics is required.

Ontologies are useful in a range of applications, where they provide a source of precisely defined terms that can be communicated across people and applications. An information system cannot be written without a commitment to a model of the relevant world – commitments to entities, properties, and relations in that world [CJ+99]. The role of ontologies is to capture domain knowledge and provide a commonly agreed upon understanding of a domain. The common vocabulary of an ontology, defining the meaning of terms and their relations, is usually organized in a taxonomy and contains modeling primitives such as concepts, relations, and axioms [HS+97]. 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.

As shown in Fig. 5.6, the DNS ontology could represent the relationship between concepts. General speaking, we could represent the semantic information by the attributes of ontology concept or the relationship between the ontology concepts. The attributes of ontology represent the internal state of the concept, while the relationship between the ontology concepts represents the outside context information of concepts. If we focus on specific domain, the ontology would provide us much background domain knowledge. First, the taxonomy hierarchy information could provide us the inheritance information. As shown in Fig. 5.6, the "is_a" relationship between DNS concepts and Master/Slave DNS concepts indicate that both master/slave DNS concepts are a kind of DNS. Second, we could define required relationship for application requirement. For example, if we need to represent synonym information, we could define the "synonym" relationship. Hence, during the ontology construction, we would take into account "synonym" relationship. For example, the "synonym" relationship indicates that "Master DNS" concept and "Primary DNS" concept are identical. As for "Related" relationship, "DNS Dynamic Update" concept and "Zone Data Protection" concepts are related to "DNS Security" concept. Therefore, when the users are interested in "DNS Security", they may be interested in "DNS Dynamic Update" concept or "Zone Data Protection" concept as well. Third, ontology could provide basic inference mechanism. The reasoning capability is useful, because the inference engine could infer more results based on known information. For example, if "is_a" relationship exists between concept ***A*** and concept ***B*** and "is_a" relationship exist between concept ***B*** and concept ***C***. We would infer that the "is_a" relationship exist between concept ***A*** and concept ***C***.

Chapter 6 System Architecture

Even though DNS is so important to network operation today, many novice DNS administrators often do not know whether their DNS servers work well. Therefore, a knowledge portal which focuses on DNS domain is required. In Chapter 6, we will describe what our DNS portal is and how it operates.

6.1 Diagnosis-Learning-Search Model

It is expected that our DNS knowledge portal could at least achieve four goals. First, for those who are lack of domain knowledge and want to build up new DNS servers, our DNS knowledge portal could provide DNS-related knowledge for them. Second, for those who want to check whether their DNS works well and do not know how to do that, our DNS knowledge portal could help diagnose their DNS servers. Third, for facilitating the reusability and interoperability, the DNS teaching materials would be wrapped by the SCORM standard. Fourth, for those who would like to search required information on the portal, we provide DNS ontology-based search service to enhance the searching capability and improve the usability of the search service.

Fig. 6.1 shows the overview of the whole system, which consists of the diagnosis service, the tutoring service and the search service. One of the key features (or requirements) of the proposed portal system is that, in addition to the individual services, the integration of all the services is important as well. For example, the DNS model-tracing tutoring could adopt the diagnosis service as the DNS knowledge test interface. On the other hand, the DNS diagnosis service could adopt the model-tracing service as the further tutoring system. And, the search service could provide the search mechanism (i.e., as for traditional searching the data records in the database, the articles in the file system or the teaching materials) to look for required information such as the most appropriate tutoring material and related configuration and design suggestions.

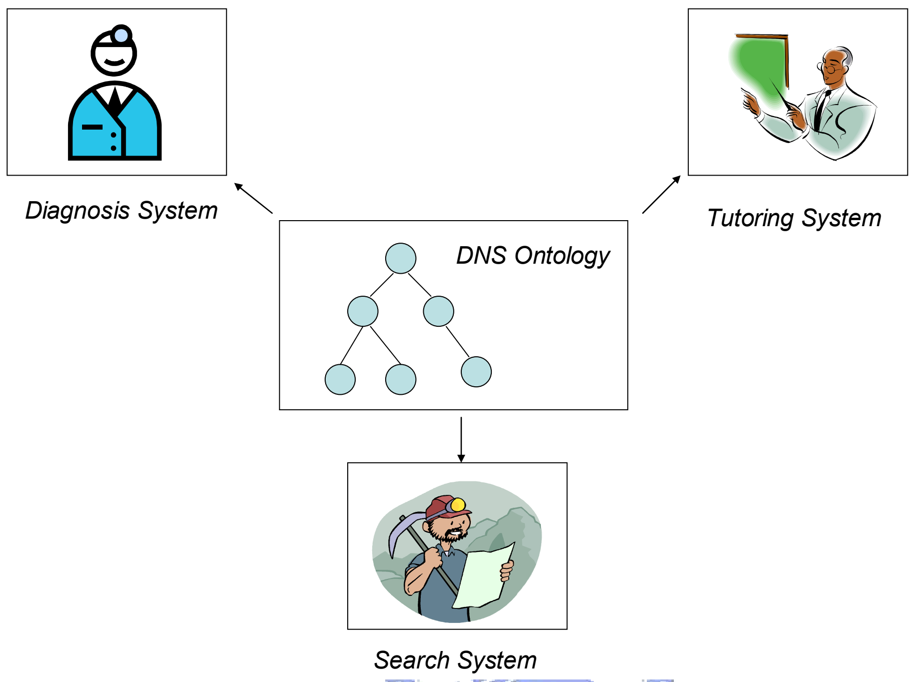


Fig. 6.1: iDNS-MS system architecture

In the following sections, we would describe DNS diagnosis service, tutoring service and search service respectively.

6.2 DNS Diagnosis System

It is expected that our DNS knowledge portal could at least achieve four goals. First, for those who are lack of domain knowledge and want to build up new DNS servers, our DNS knowledge portal could provide DNS-related knowledge for them. Second, for those who want to check whether their DNS works well and do not know how to do that, our DNS knowledge portal could help diagnose their DNS servers. Third, for facilitating the reusability and interoperability, the DNS teaching materials would be wrapped by the SCORM standard. Fourth, for those who would like to search required information on the portal, we provide DNS ontology-based search service to enhance the searching capability and improve the usability of the search service.

Fig. 6.1 shows the overview of the whole system, which consists of the diagnosis service, the tutoring service and the search service. One of the key features (or requirements) of the proposed portal system is that, in addition to the individual services, the integration of all the services is important as well. For example, the DNS model-tracing tutoring could adopt the diagnosis service as the DNS knowledge test interface. On the other hand, the DNS diagnosis service could adopt the model-tracing service as the further tutoring system. And, the search service could provide the search mechanism (i.e., as for traditional searching the data records in the database, the articles in the file system or the teaching materials) to look for required information such as the most appropriate tutoring material and related configuration and design suggestions.

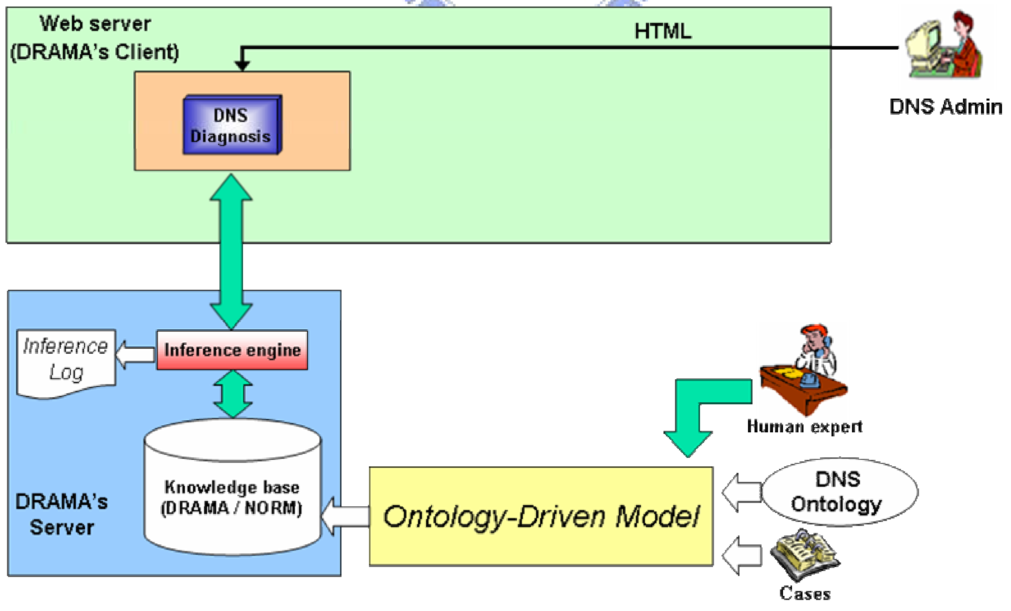


Fig. 6.2: System architecture of DNS diagnosis system

In traditional rule-base expert system, the rule base 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 knowledge class to manipulate the knowledge and loads only the required knowledge classes. That could simplify the rules management and improve the efficiency of the KBS. In essence, each knowledge module is corresponding to the knowledge class (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 rule base, which facilitates the management of rules in each knowledge class. 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 knowledge classes takes into account knowledge reuse. For example, the rule,

*If **TTL1** != **TTL2** then **LameServer** = true*

, is located in "DNS Registration" knowledge class and it needs the facts of DNS server knowledge class. In principle, the facts "***TTL1***" and "***TTL2***" in the "DNS Server" KC will be taken (transferred) to the "DNS Registration" KC. Therefore, as shown in Fig. 6.3, there is a relation "**Acquire**" between them.

- Rule ：*If **TTL1** != **TTL2** then **LameServer** = true*

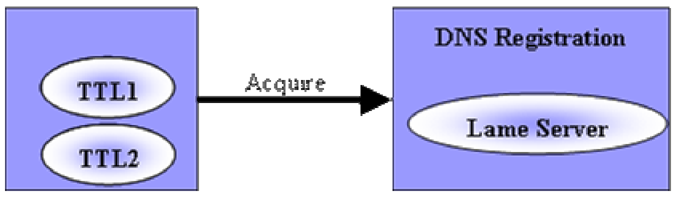


Fig. 6.3: "DNS Registration" KC acquires the facts of "DNS server" KC

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 the information about the DNS environment in more detail, the questions and answer model is used to help acquire the user's DNS information.

In addition, we adopt Model-View-Controller design pattern [KP98] 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.

6.3 DNS Ontology-Based Model-Tracing Tutoring

In DNS diagnosis system, the system will provide the suggestions when something wrong with users' DNS configurations. However, the suggestions could only reflect the actions required to fix the problems and some of the users will not really understand the reasons. Therefore, we start to think about the integration model of diagnosis system and tutoring system. As described above, the DNS domain is problem driven and most of the DNS administrators are interested in the topics that are related to their DNS configuration errors. Therefore, for those who would like to know the DNS operation model in more detail, the system could provide more tutoring teaching materials. In addition, the diagnosis system could diagnose users' DNS configurations and that could be viewed as users' DNS configuration behaviors. Fig. 6.4 shows the proposed architecture of the integrated DNS tutoring/diagnosis system. As described above, in diagnosis system, we propose an ontology-driven model for rule extraction and store the knowledge into the KBS (***DRAMA/NORM***). In previous design [LT+04-1], these collected facts will be sent to the inference engine and then the inference results will return to the web interface. As we all know, the more detailed information is collected, the more accurate suggestions could be provided. To make the system more complete, we further refine the working paradigm. When the users finish the diagnosis processes, the diagnosis system will return some suggestions about their DNS hosts. If any users would like to know more about their problems, they could start the DNS tutoring process based on the firing rules (in the diagnosis results) to learn more about their DNS systems and related problems.

In short, DNS tutoring system, based on ontology-based model-tracing tutoring model, will provide users appropriate teaching materials or online help when receiving inference result (DNS problem and firing rules) from diagnosis system. In addition, on specific conditions (such as lame servers and SPOF) when the users finish some tutoring courses, DNS tutoring system might ask the users to reconfigure their own DNS hosts again and start another diagnosis. The new diagnosis result will be used to analyze whether the users understand the courses.

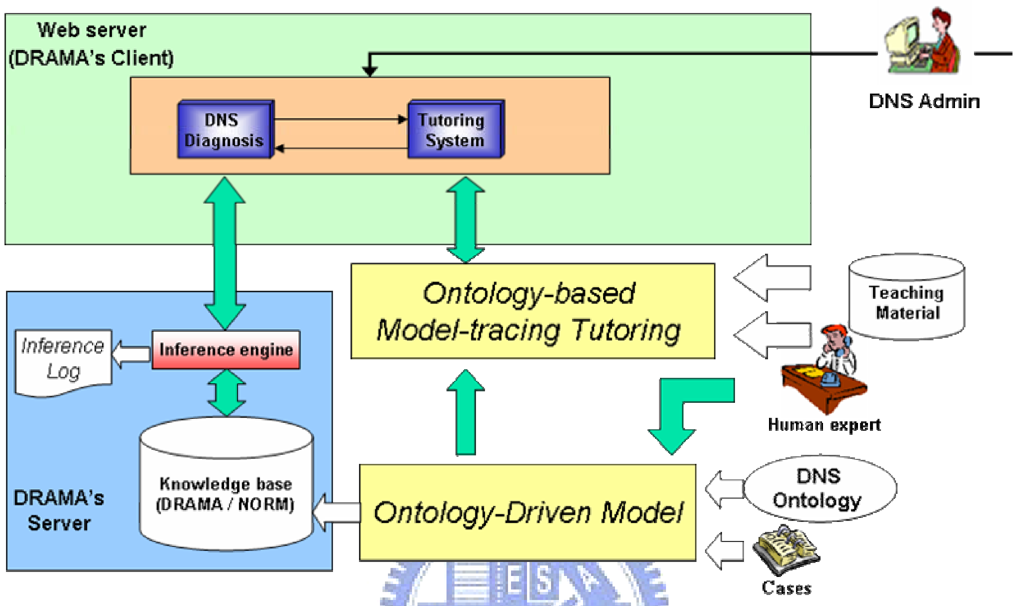


Fig. 6.4: System architecture of DNS tutoring system

In traditional tutoring system, the teaching materials are arranged by chapters and the students usually learn the topics in the listed order sequentially. In a sense, the chapter structure would represent learning paths for the course. For example, algebra symbolization should be introduced before learning mathematics equations and the students would learn algebra symbolization before mathematics equations. However, in general, the chapter-structure representation of DNS domain knowledge might not be a good enough way to provide DNS learning for many people (i.e., especially for the inexperience DNS administrators) to deal with the complicated internetworking environment for several reasons such as timing issue and the complexity of the knowledge. In other words, many DNS administrators usually attempt to know the appropriate topics related to the problems of their DNS servers in a timely manner.

Fig. 6.6 shows a reference hierarchy of DNS tutoring materials, collected from the reference materials from domain experts and DNS-related books. For example, introduction to DNS issues is the basis of DNS tutoring. It includes DNS terminology, concepts, operations, etc. All of the other DNS issues except DNS introduction could be viewed as independent courses and will refer to DNS introduction issue and other DNS-related or network-related issues if needed.

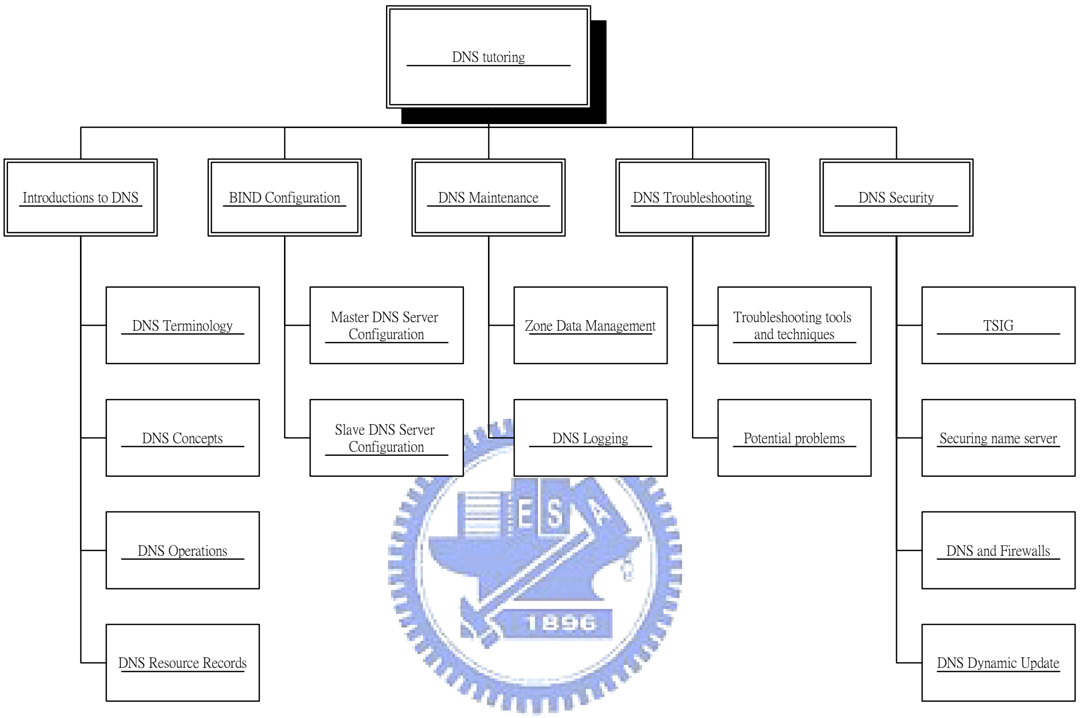


Fig. 6.5: DNS tutoring teaching material hierarchy

In practice, during the tutoring process, it will be of great help for the system to provide appropriate auxiliary mechanisms to assist the users to learn the course more smoothly. Furthermore, in addition to the online help, the content and ways of presentation of the teaching materials are important as well. Since the model-tracing tutoring skeleton could provide possible problem-solving paths about the diagnosis/tutoring process and the diagnosis rules could reflect users' activities, the domain experts could provide appropriate assistances more easily at proper time. However, as to the design and arrangement of teaching materials, it is not easy to provide them directly without domain experts' help. For example, according to our experiences, the tree representation is more easily understood than many other ways. After we finished transforming the knowledge embedded in the ontology into the model-tracing tutor skeleton, the domain experts could provide appropriate teaching materials on the tree nodes to facilitate the acquisition and growth of more knowledge objects (i.e., course materials) on top of the skeleton.

Furthermore, the association relationship between the teaching materials is important as well. For some cases, the specific administrators need only to know the issues about their DNS hosts, but others may need more. For example, when dealing with DNS SPOF problem, we might have to check the DNS NS resource records, master DNS server, slave DNS server and network-related issues. Therefore, instead of providing all related materials once, it is better to provide teaching materials incrementally. On considering these, we interview the domain experts to build the connections between the rules node as shown in Fig. 5.5 and teaching materials as shown in Fig. 6.5. That is, we interview DNS domain experts for acquiring the knowledge (and the relationships) among the required teaching materials based on the pseudo rules and present the topics incrementally. For example, Table 6.1 shows the required teaching material for DNS SPOF pseudo rules. When the users' inference results fire the rule "number of NS records < 2", the DNS tutoring system will provide "DNS operations" course first to them. After finishing "DNS operations" course, the system will ask the users if they could manage to reconfigure their own DNS server and start another diagnosis test after the reconfiguration is done.

Table 6.1: Teaching materials for specific pseudo-rules

Pseudo rule	Suggestion in Diagnosis System	Teaching Material
Number of NS	At least two NS records (Master	• DNS Operations

Record <2	DNS server and Slave DNS server) are required.	• DNS Resource Records • DNS Concepts • DNS Terminology • Master DNS Configuration • Slave DNS Configuration
Master DNS and Slave DNS are not live	Each zone should have one and only one master DNS server. Each zone should have at least one slave DNS server (and may be more).	• Master DNS Configuration • Slave DNS Configuration • DNS Operations • DNS Resource Records • DNS Concepts
Master DNS Server and slave DNS server are in the same network location	DNS servers should be located in different network location	• DNS Operations • DNS Concepts
The location of Master and Slave DNS servers are behind the same router	DNS servers should not located behind the same router	• DNS Operations • DNS Concepts

The next step will be based on the diagnosis test. If the users could not pass the tests, in addition to "DNS operations" course, the system will provide "DNS Resource Records" course as well. Instead of telling the users how to do it directly, we would like to guide the users and let the users do it themselves. Therefore, the users might need to re-configure their DNS to see whether they understand the course. Fig. 6.6

shows the flow of DNS tutoring process, which adopts the DNS diagnosis subsystem as the testing environment, the DNS tutoring subsystem as tutoring environment and model-tracing model as the medium for connecting the two subsystems.

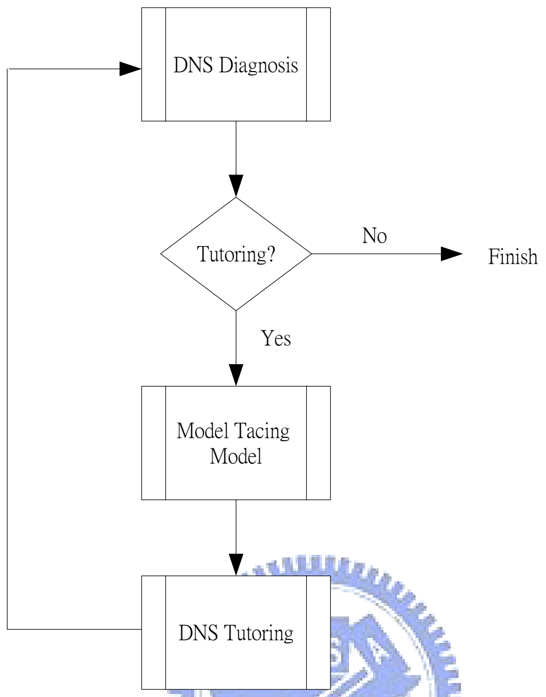


Fig. 6.6: DNS tutoring flow

6.4 DNS Ontology-Based Searching

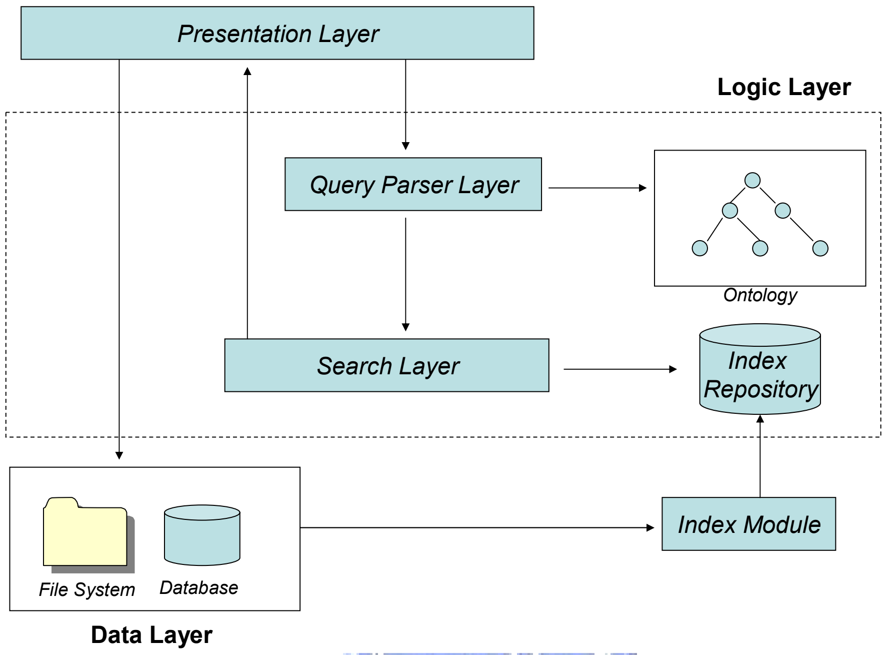


Fig. 6.7: System architecture of ontology-based search system

1896

In addition to diagnosis system and tutoring system, there are many DNS related articles or information in the system. In the traditional information system, search mechanism is the basic tool for the information search. Therefore, search is often the core of the existing portal systems (e.g. yahoo, MSN, pc home etc.). However, most of the existing search systems are based on keyword search and that may lead to incorrect results. Some portal systems (such as yahoo) provide the directory taxonomy information during the search process. The taxonomy information could provide much help. For example, Fig. 6.8 shows the AI taxonomy hierarchy information and we could infer that the expert system, GA, neural network and fuzzy all belong to AI field. Furthermore, if the search engine posses the knowledge, the search engine would be more intelligent. For example, when someone search the term “AI”, the search engine could provide the category information and could provide more suggested search query term (such as, expert system, neural network, GA, fuzzy) based on the taxonomy information. However, it is not easy to construct the general purpose taxonomy hierarchy but for specific domain, the taxonomy hierarchy structure would be possible.

Ontology could provide the basic taxonomy hierarchy information. In addition, the ontology could provide inference mechanism for further information reasoning and that could improve the capability of search system. In addition, the flexibility of the search system is important as well. Now the data source of the system includes the articles in the file system, the data record in the database. The new data source (e.g. mailing list archie, news, blog etc.) may be taken into account in the future. Furthermore, a scalable system which could provide robust services is important as well. Therefore, in the system design, we take into account these issues and propose a three-layer framework. As shown in Fig. 6.7, the whole system could be divided into presentation layer, logic layer and data layer respectively. The descriptions of the layers are listed as follows:

- Presentation Layer: The presentation layer focuses on the user interface and the search result presentation. When the user enters the search keyword and criteria, the presentation layer will collect and pass the information to the java servlet for further processing.
- Logic layer: The role of logic layer is to act as the bridge between presentation layer and data layer and the logic layer could be further divided into two sub-layers, query parser layer and search layer. The logic layer will receive the query input from presentation layer and the query parser layer will trigger the internal inference engine based on DNS ontology. The inference result will then pass to the search layer and start the search process. The search layer will search the index repository for the required information and return to the presentation

layer.

- Data layer: The data layer contains different data sources (e.g., the files in the file system, the data record in the database etc.). To speed up the search process, it is necessary to index these data. In addition, the design of data layer should take into account into the flexibility. For example, when different data sources (e.g., the mailing list archie, the pages in the Internet etc.) are imported, the data layer should be able to handle the new data source without changing the existing design.

Just like the MVC design pattern, our design is focus on the separation of the presentation, logic processing and data. Therefore, if we would like to change the design of arbitrary layer, we need not change the design of the other layers. For example, if we add more data sources into the data layer, the logic layer still access the result of the index and the presentation layer presents the result as usual. Furthermore, the way of domain ontology representation may vary. For example, we could represent the ontology by using XML, RDF, or OWL etc. Different ontology representations need different process logic. Therefore, to improve the flexibility of the ontology representation, we use the java interface design to abstract the inference engine design.

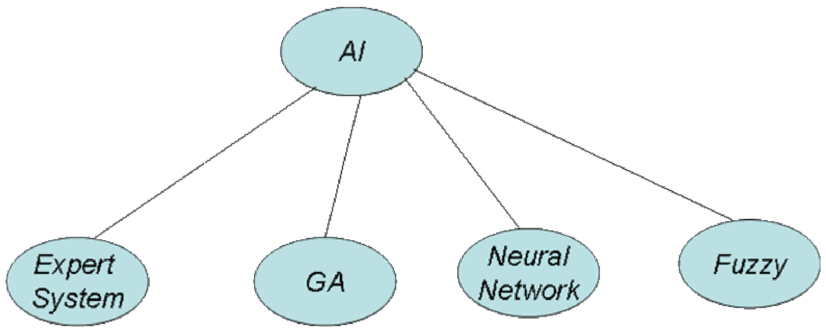


Fig. 6.8: The AI hierarchy

Fig. 6.9 shows the ontology inference engine design flow. The input is the query string from users. The inference engine could support many ontology representation formats when we provide the required implementations. Interface and implementations are separated and that could improve the reusability of the system. In addition, we adopt Apache ***Lucene*** as the search framework, so we need to translate the inference result into the format which is acceptable by ***Lucene***.

Lucene is a high-performance and scalable search engine technology. The powerful abstractions and useful concrete implementations make ***Lucene*** very flexible. It provides the basic search architecture and it has been applied on many domains. As for the search section, ***Lucene*** provides the query parser mechanisms that could fulfill most of your requirements. For example, most of the search engines provide the *Boolean* mechanism for the users to composite complex query. If you would like to search the documents containing "***DNS***" and "***Linux***" but not "***Windows***", you could use the following query string:

DNS AND Linux Not Windows

In addition to ***Boolean*** query parser, ***Lucene*** provides other query parser mechanisms (e.g. Term Query, Fuzzy Query, Wildcard Query etc.). In our system design, we make use of the ***Lucene*** query parser mechanisms to represent the final inference result.

Furthermore, in our search system, we adopt XML as the ontology representation format. Fig. 6.10 shows part of the DNS ontology. In the XML file, all the concepts are represented by "class" tag. The concept could consist of property attribute. In addition, we could define the relationship between the concepts. For example, we could define synonym relationship between "***Primary DNS***" and "***Master DNS***" concepts. When the inference engine parses the XML file, it would reason that "***Primary DNS***" and "***Master DNS***" are identical, and they should be taken into account at the same time. Therefore, the inference engine would transfer the original query string "***Master DNS***" into "***(Master DNS OR Primary DNS)***". In addition to synonym relationship, we define the "***Related***" relationship. The related relationship could be used to model the general terms condition. For example, if the users would like to find out the documents about DNS security issues. Although DNS security consists of many other related issues (e.g. ***DNS Spoofing***, ***Zone Data Protection*** etc.), most of the users do not know these detail issues. In most of the traditional search systems, the users may miss some information. Therefore, in our system, we define the ***Related*** relationship to solve this kind of problem.

In the data layer design, we need to take into account the possibility of new data source requirement. When the new data source comes, the system should not change the original design. To facilitate the communication of programmer and system analyzer, we adopt Unified Modeling Language (abbreviated as UML [Kobryn99]) as the visualizing, construction and documenting language. Fig. 6.11 shows the class diagram of the index class design. Since we may face different data source, we adopt "Factory" design patter to achieve the goal. The *IndexFactory* is similar to a factory which is used to create different index sources. The *Factory* design pattern could hide the detail implementation from the clients. Therefore, if new data source comes, the clients do not change. In addition, to separate the implementation from the design, the *IndexSource* interface defines the required method *addDocuments* for every concrete class that implement *IndexSource* interface. Therefore, if we need to add a new data source, we could follow the following steps:

1. Create a concrete class which implement *IndexSource* interface.
2. Fill in the required *addDocuments* method.

For example, if we would like to process HTML files, we could create a *FSHTMLIndexSource* class which implements *IndexSource* interface. In the *addDocuments* method, we would need to parse the HTML information first and then extract the required information (e.g. title, body etc.) and composite these information as *Lucene Document* object.

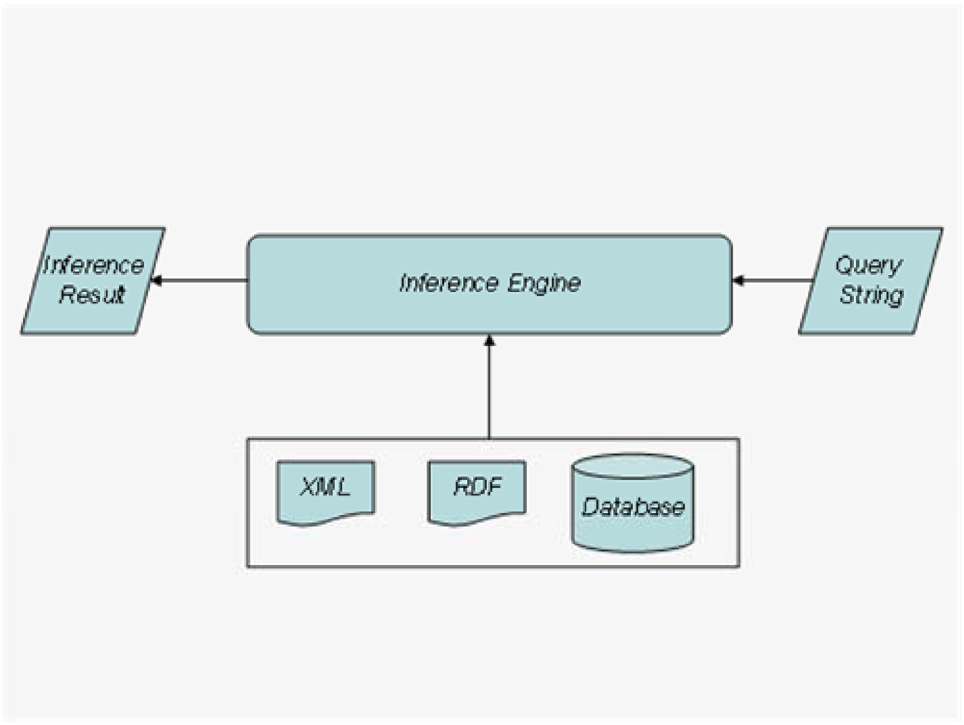


Fig: 6.9: Ontology inference engine flow

```
<Ontology>
    <Class name="DNS">
        <Synonym name="Master DNS"/>
        <Property name="SOA"/>
        <Property name="NS"/>
        <property name="MX"/>
    </Class>

    <Class name="Primary DNS" >
        <Synonym name="Master DNS"/>
    </Class>

    <Class name="DNS Security">
        <Related name="DNS Spoofing"/>
        <Related name="Zone Data Protection"/>
    </Class>

    …


</Ontology>
```

Fig. 6.10: DNS ontology XML

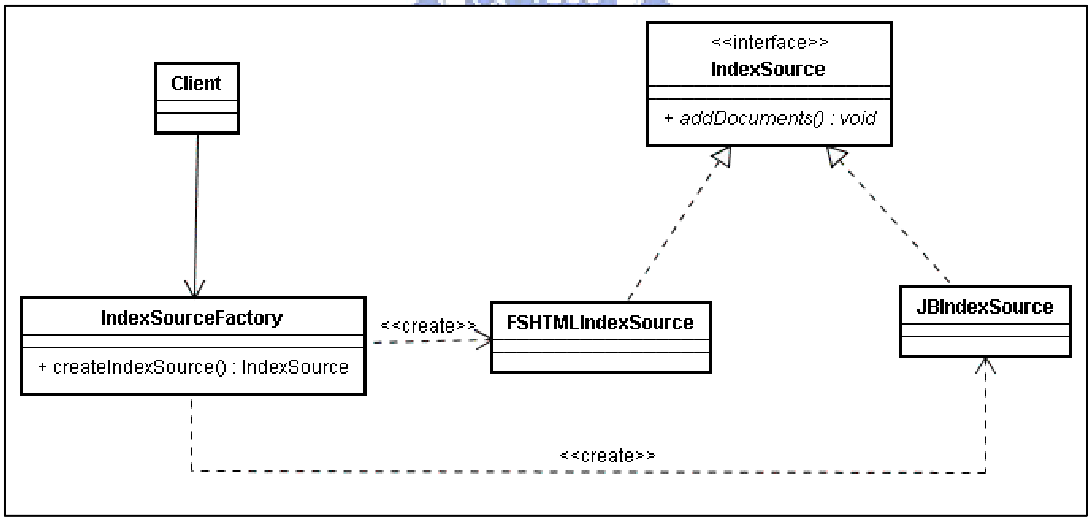


Fig: 6.11: Index class diagram

Chapter 7 Implementation and Evaluation

7.1 System implementation

As with the popularity of Internet, web application has become one of the most popular application models and most of the people are familiar with the web interface. In addition, we take into account the system portability issue as well. When the number of users grows, we may need to move the existing system to different environment. In general, the following issues are required during the design:

1. Robust issue:

The robust issue is the most important issue. When we provide the services, we hope the user could access the services without any problem. 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 [Wu00]. ***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.

2. Portability issue::

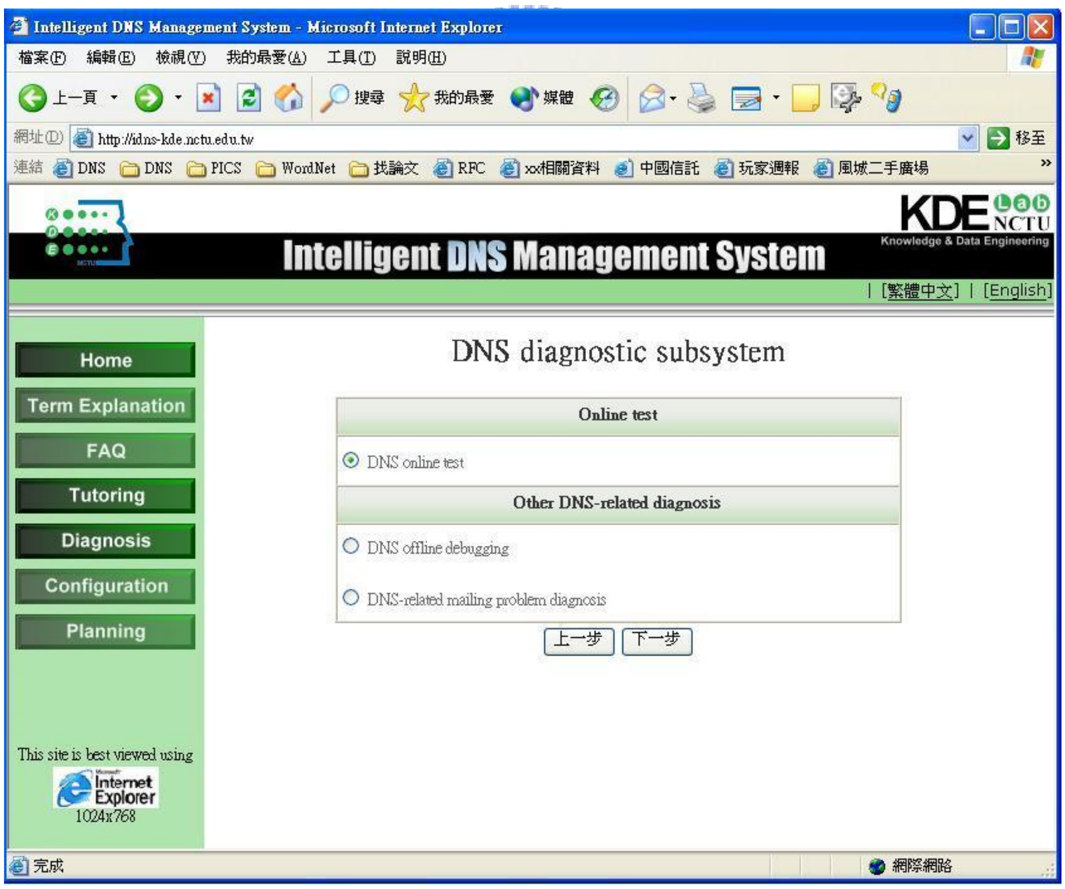
To improve the system portability of the system, we adopt JAVA as the implementation platform. Therefore, if we would like to change the OS, we do not need to change the code.

3. Standard issue:

The standard issue is important as well. If we follow the standard, when we need to exchange information with other system, the burden will be low. Therefore, we adopt SCORM standard as the tutoring platform.

In addition, open source software plays an important role in our system implementation. 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. In the following sections, we would describe the diagnosis service example in Section 7.2, model-tracing tutoring service example in Section 7.3, building an ENUM DNS example in Section 7.4, DNS ontology-based search service example in Section 7.5 and finally the evaluation in Section 7.6.

7.2 Diagnosis examples



Intelligent DNS Management System - Microsoft Internet Explorer
檔案(F) 編輯(E) 檢視(V) 我的最愛(A) 工具(T) 說明(H)
上一頁 搜尋 我的最愛 媒體
網址(D) http://idns-kde.nctu.edu.tw
移至
連結 DNS DNS PICS WordNet 找論文 RFC xx相關資料 中國信託 玩家週報 風城二手廣場
KDE Lab NCTU
Knowledge & Data Engineering
Intelligent DNS Management System
[繁體中文] [English]
Home
Term Explanation
FAQ
Tutoring
Diagnosis
Configuration
Planning
DNS diagnostic subsystem
Online test
DNS online test
Other DNS-related diagnosis
DNS offline debugging
DNS-related mailing problem diagnosis
上一步 下一步
This site is best viewed using Internet Explorer 1024x768
完成
網際網路

Fig. 7.1: The DNS diagnostic subsystem

As shown in Fig. 7.1, there are three diagnosis facilities for users to choose:

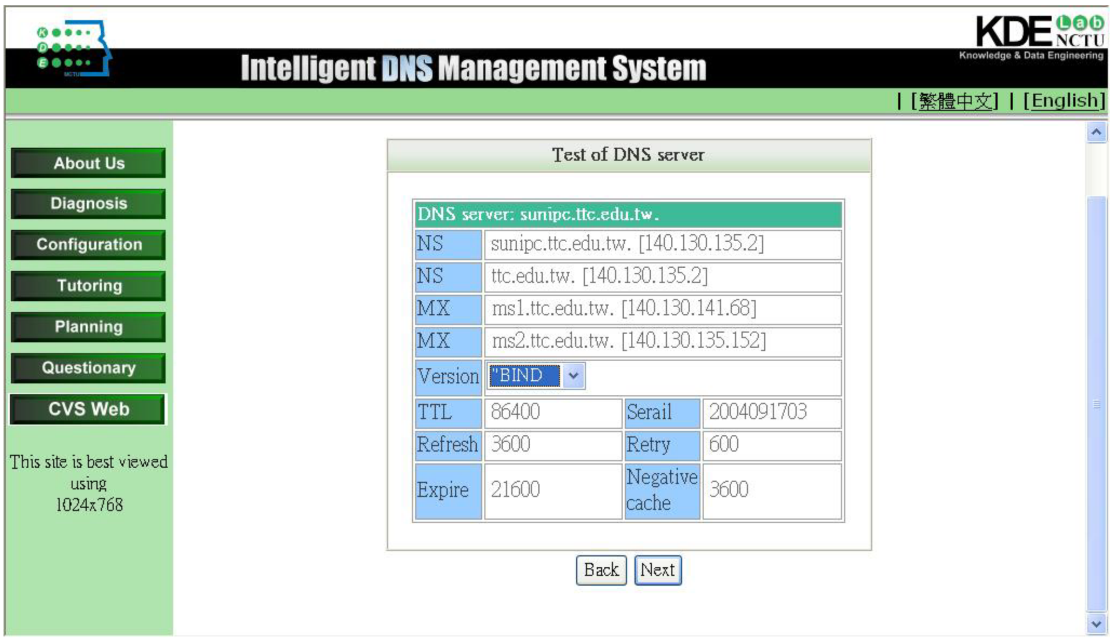
- DNS on-line test：This will test the DNS servers that are supposed to be responsible for the domain zone. All users need to do is to enter a domain name and to select the DNS server. Then our system will conduct the required DNS queries and collect the information about this server from Internet automatically. 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.

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. 7.1, 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***" knowledge class is triggered. Next, according to the cases, our system will further try to identify 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).

Next, as shown in Fig. 7.2, the system will start to collect facts for later inference by asking the users to enter the domain zone name (about their DNS servers) wanted. Based on the users' input information, the DNS diagnosis system then start another private session(s) to access the DNS servers, regarding the domain zone under test, for more facts (as shown in Fig. 12) and start the diagnosis process.

As shown in Fig. 5.3, 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" knowledge class. 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. 7.3.

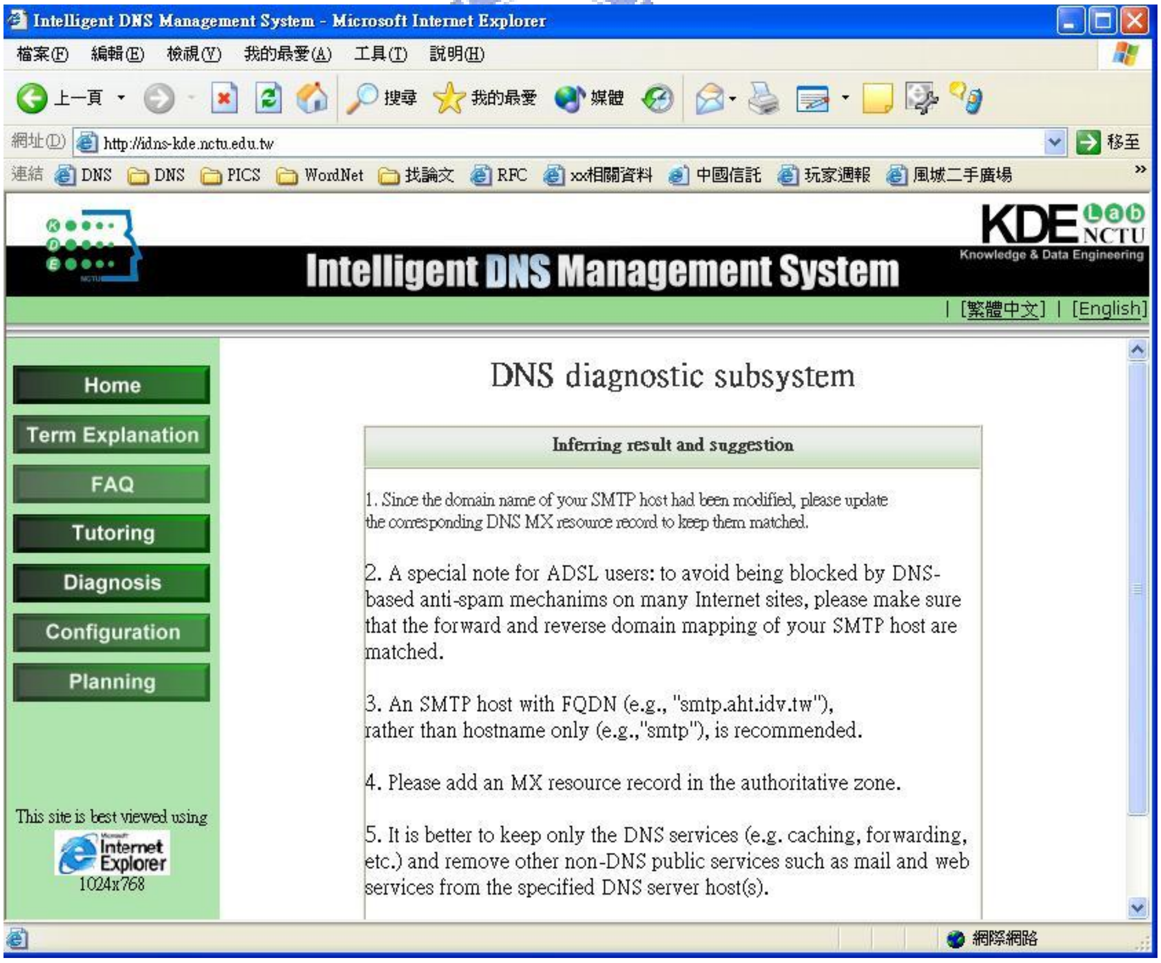


Intelligent DNS Management System
KDE Lab NCTU Knowledge & Data Engineering
| [繁體中文] | [English]
About Us
Diagnosis
Configuration
Tutoring
Planning
Questionary
CVS Web
This site is best viewed using 1024x768
Back
Next

Test of DNS server

DNS server: sunipc.ttc.edu.tw.			
NS	sunipc.ttc.edu.tw. [140.130.135.2]		
NS	ttc.edu.tw. [140.130.135.2]		
MX	ms1.ttc.edu.tw. [140.130.141.68]		
MX	ms2.ttc.edu.tw. [140.130.135.152]		
Version	"BIND		
TTL	86400	Serail	2004091703
Refresh	3600	Retry	600
Expire	21600	Negative cache	3600

Fig. 7.2: DNS testing on DNS diagnosis system



Intelligent DNS Management System - Microsoft Internet Explorer
檔案(F) 編輯(E) 檢視(V) 我的最愛(A) 工具(T) 說明(H)
上一頁 搜尋 我的最愛 媒體
網址(D) http://idns-kde.nctu.edu.tw 移至
連結 DNS DNS PICS WordNet 找論文 RFC xx相關資料 中國信託 玩家週報 風城二手廣場
Intelligent DNS Management System
KDE Lab NCTU Knowledge & Data Engineering
| [繁體中文] | [English]
Home
Term Explanation
FAQ
Tutoring
Diagnosis
Configuration
Planning
This site is best viewed using Internet Explorer 1024x768
網際網路

DNS diagnostic subsystem

Inferring result and suggestion

1. Since the domain name of your SMTP host had been modified, please update the corresponding DNS MX resource record to keep them matched.

2. A special note for ADSL users: to avoid being blocked by DNS-based anti-spam mechanims on many Internet sites, please make sure that the forward and reverse domain mapping of your SMTP host are matched.

3. An SMTP host with FQDN (e.g., "smtp.aht.idv.tw"), rather than hostname only (e.g.,"smtp"), is recommended.

4. Please add an MX resource record in the authoritative zone.

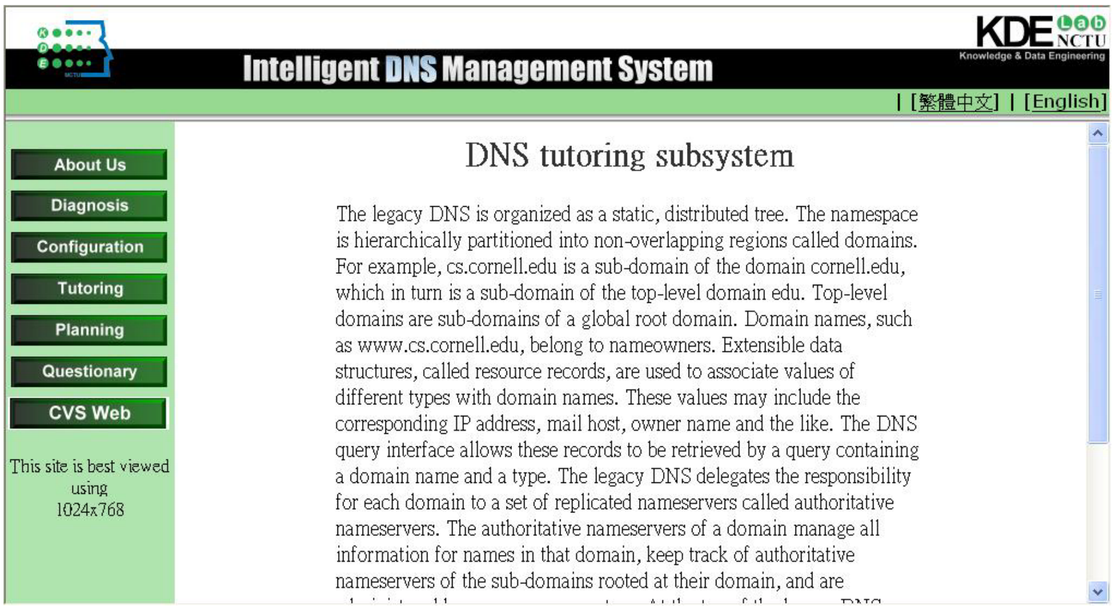
5. It is better to keep only the DNS services (e.g. caching, forwarding, etc.) and remove other non-DNS public services such as mail and web services from the specified DNS server host(s).

Fig. 7.3: Inference results of DNS-related mail delivery problems

7.3 Model Tracing Tutoring examples

According to domain expertise, DNS Single-Point-of-Failure (abbreviated as SPOF-DNS) problem is one of the most common problems in DNS deployment on Internet sites. In practice, however, due to the lack of experience and domain knowledge, many inexperienced DNS administrators did not realize these might become critical problems under specific network situations sometime in the future. In principle, it will be an obvious weak point that abusers (or attackers) could exploit to break the availability of the network services of the target site since DNS is the infrastructure of Internet.

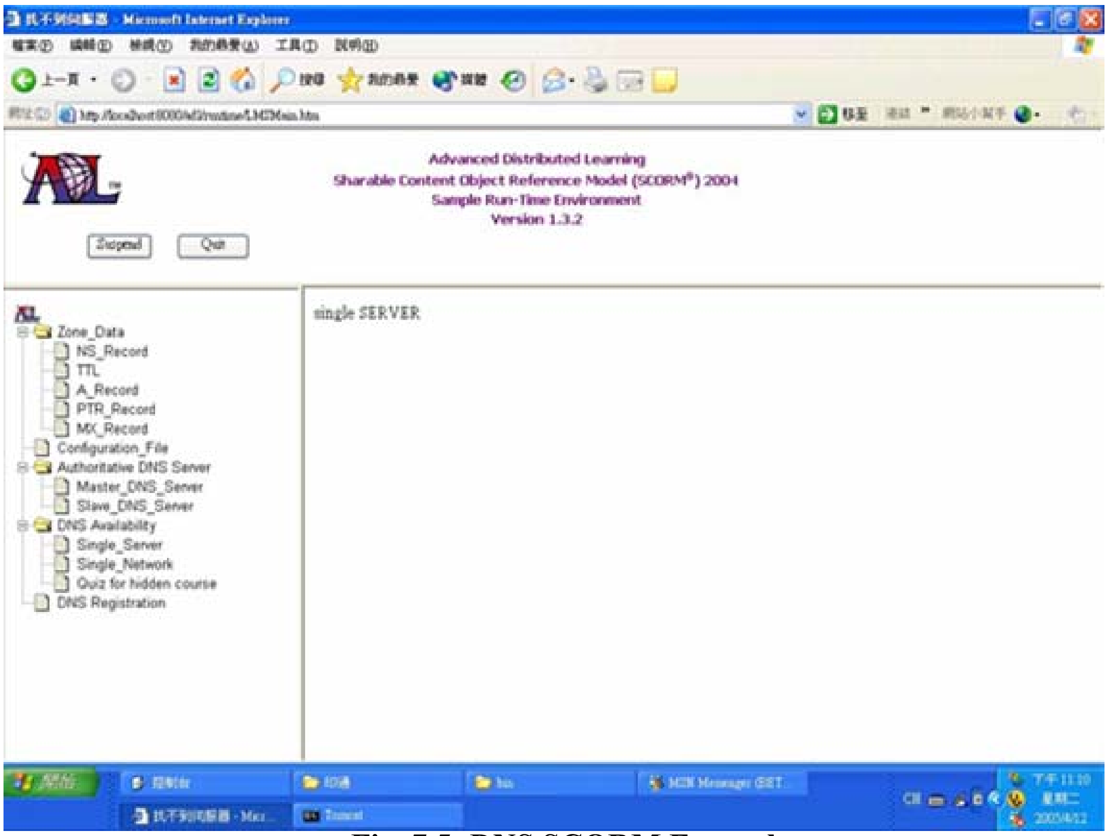
In our previous DNS diagnosis work, users can use the diagnosis system to get plausible solutions on various types of DNS-related diagnosis services. After finishing diagnosing, if the users would like to know more about the specific problem or DNS operation principle, the DNS tutoring system would give the users appropriate tutoring teaching materials. As shown in Fig. 7.4, the diagnosis system could retrieve users' DNS configuration information as shown in Fig. 7.2 as users' DNS activities and the production rules extracting in the process of ontology-driven rules extraction model to simulate users' behavior. In addition, as shown in Fig. 7.5, based on the learning sequences construction using ontology and rules algorithm, we build the DNS SCORM learning environment. That could provide the users another way to learn DNS knowledge.


Intelligent DNS Management System
KDE Lab NCTU
Knowledge & Data Engineering
| [繁體中文] | [English]
About Us
Diagnosis
Configuration
Tutoring
Planning
Questionary
CVS Web
This site is best viewed using 1024x768

DNS tutoring subsystem

The legacy DNS is organized as a static, distributed tree. The namespace is hierarchically partitioned into non-overlapping regions called domains. For example, cs.cornell.edu is a sub-domain of the domain cornell.edu, which in turn is a sub-domain of the top-level domain edu. Top-level domains are sub-domains of a global root domain. Domain names, such as www.cs.cornell.edu, belong to nameowners. Extensible data structures, called resource records, are used to associate values of different types with domain names. These values may include the corresponding IP address, mail host, owner name and the like. The DNS query interface allows these records to be retrieved by a query containing a domain name and a type. The legacy DNS delegates the responsibility for each domain to a set of replicated nameservers called authoritative nameservers. The authoritative nameservers of a domain manage all information for names in that domain, keep track of authoritative nameservers of the sub-domains rooted at their domain, and are

Fig. 7.4: DNS tutoring subsystem



single SERVER

Fig. 7.5: DNS SCORM Example

7.4 Building an ENUM DNS server

In practice, it might be hard for most administrators to deal with the construction and management of ENUM DNS systems since it involves both the complex DNS and new ENUM protocol suites. In this thesis, DNS knowledge portal is extended to help deal with these issues. It is supposed that not only could the DNS knowledge portal help some users solve their ENUM DNS deployment problems, but it also provides other administrators with the insight about how to design and implement their ENUM DNS systems. For example, if the administrators could have a deeper understanding about these management issues in advance through some subsystems such as the DNS term explanation and tutoring, the probability of making mistakes will be lower in the future. Similarly, DNS design subsystem could give the administrators some suggestions when the environment of ENUM DNS (number of user, network topology etc.) changes. Meanwhile, DNS diagnosis and configuration subsystems could help the user to debug and find the solutions.

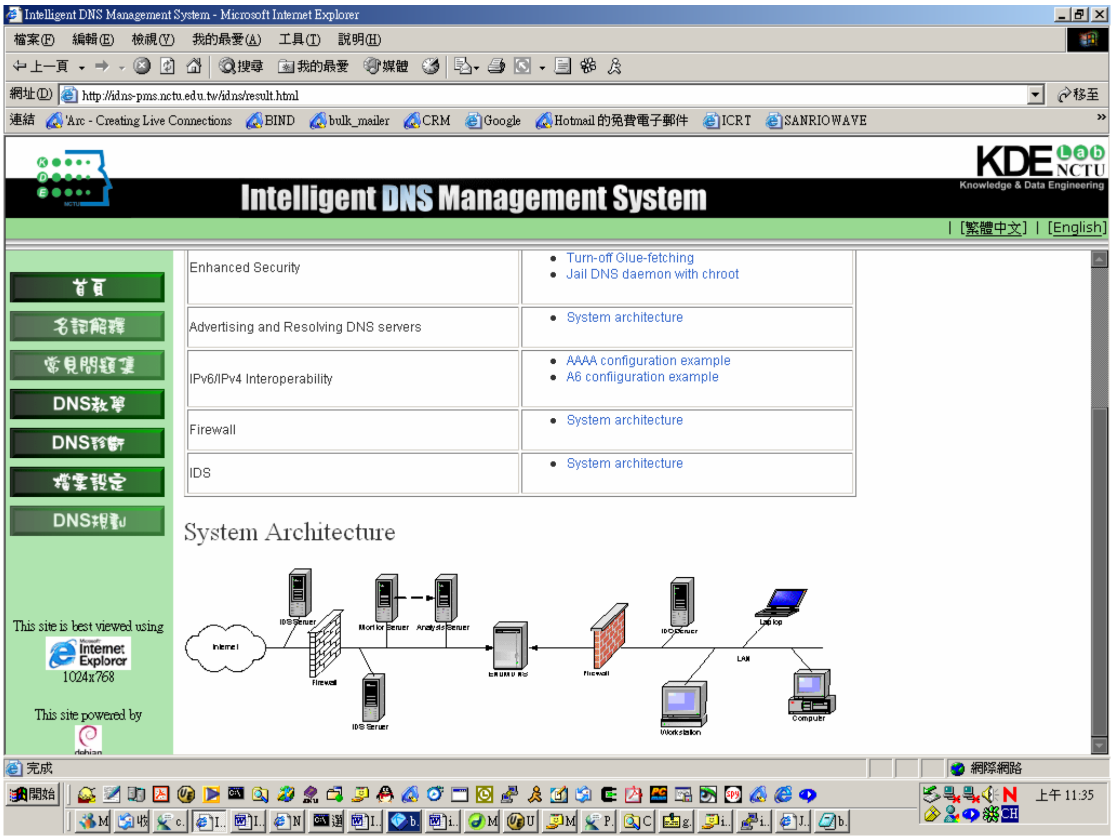
As shown in Table 7.1, many management issues need considering during the construction of ENUM DNS. First, because the IPv6 infrastructure on most sites is still under construction, it might result in many new DNS problems. Second, since ENUM DNS is a special kind of DNS, it needs many protection mechanisms (e.g., thorough planning, configuration, monitoring, diagnosis, etc.) to ensure that it operates well.

Table 7.1: ENUM DNS properties

Problem/Issues	Protection mechanisms
DDos attack/ SPAM Mail attack	■ System monitor ■ Network software or hardware ■ IDS ■ Firewall
Availability	■ Eliminate SPOF (Single Point Of Failure)

	■ Keep the DNS server simple and light ■ Exclude all Internet services (e.g. WWW, proxy, ftp, etc.) that are not necessary for conducting DNS services on the DNS server host. ■ Separation of DNS traffic ■ Advertising server : incoming ■ Resolving server: outgoing
Security	■ Restrict zone transfer ■ TSIG ■ DNSSEC ■ Avoid dynamic update ■ Avoid DNS spoofing ■ Turn-off recursive query ■ Turn-off glue-fetching ■ Jail DNS daemon with chroot
IPv6/IPv4 Interoperability	■ Dual-stack DNS

Our system will base on the inputs from users and then the inference engine will return the results to the users. We follow the object-oriented programming (OOP) approach to design ENUM DNS KBS. Basically, the whole ENUM DNS could be viewed as an object, which inherits both IPv6 and IPv4 DNS objects. The properties of ENUM DNS could be viewed as the attributes of OOP and the ENUM DNS architecture as the method of OOP. As shown in Table 3.1, some general issues will trigger the IPv4 DNS rules. For example, the availability property ("Eliminate SPOF") will trigger the IPv4 DNS rules to eliminate SPOF problem. On the other hand, IPv6/IPv4 interoperability property is valid in both IPv6 and ENUM DNS, but absent in IPv4 DNS. So, an ENUM DNS should trigger the rules in IPv6 DNS. These rules imply that it is better for a dual-stack host to have separate names. For example, the name "www.test.com" refers to the IPv4 address and its counterpart "www.ipv6.test.com" has IPv6 address. If configured in this way, we could help reduce the possibility for the DNS to respond with invalid information for the clients to access the corresponding remote system.



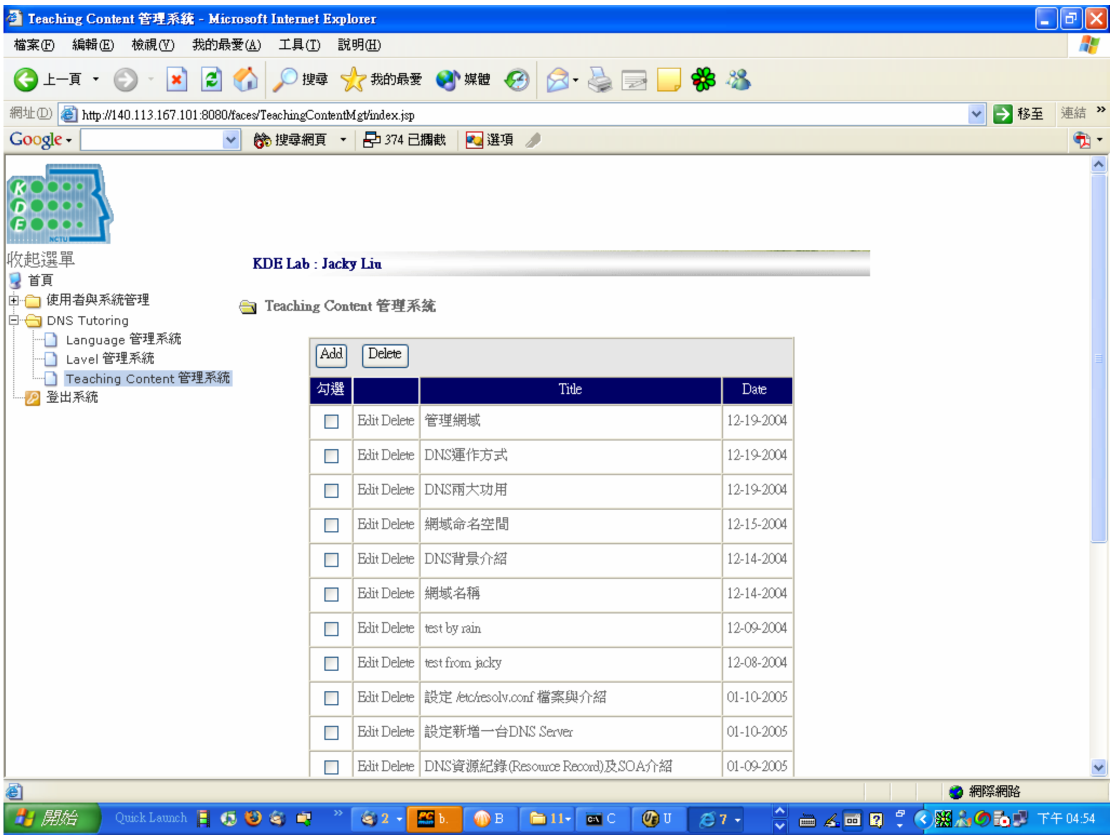
Enhanced Security	• Turn-off Glue-fetching • Jail DNS daemon with chroot
Advertising and Resolving DNS servers	• System architecture
IPv6/IPv4 Interoperability	• AAAA configuration example • A6 configiguration example
Firewall	• System architecture
IDS	• System architecture

System Architecture

Fig. 7.6: ENUM DNS result page

Fig. 7.6 shows the result page which contains a typical architecture for implementing ENUM DNS, which includes firewall, IDS (Intrusion Detection Server), monitor server and analysis server. The firewall will block all other unnecessary packets. If the DNS traffic from specific IP address is more than the threshold, the monitor server will trigger the analysis server to start dumping and analyze the traffic from the IP address. If some attacks are identified, new filtering rules will be generated by the analysis server and forwarded to the firewall to block the traffic from the IP address or traffic shaping for the IP address.

7.5 DNS Ontology-based Search Service



Teaching Content 管理系統 - Microsoft Internet Explorer
http://140.113.167.101:8080/faces/TeachingContentMgt/index.jsp
收起選單
首頁
使用者與系統管理
DNS Tutoring
Language 管理系統
Lavel 管理系統
Teaching Content 管理系統
登出系統

KDE Lab : Jacky Liu

Teaching Content 管理系統

Add Delete

勾選		Title	Date
☐	Edit Delete	管理網域	12-19-2004
☐	Edit Delete	DNS運作方式	12-19-2004
☐	Edit Delete	DNS兩大功用	12-19-2004
☐	Edit Delete	網域命名空間	12-15-2004
☐	Edit Delete	DNS背景介紹	12-14-2004
☐	Edit Delete	網域名稱	12-14-2004
☐	Edit Delete	test by rain	12-09-2004
☐	Edit Delete	test from jacky	12-08-2004
☐	Edit Delete	設定 /etc/resolv.conf 檔案與介紹	01-10-2005
☐	Edit Delete	設定新增一台DNS Server	01-10-2005
☐	Edit Delete	DNS資源紀錄(Resource Record)及SOA介紹	01-09-2005

Fig. 7.7: DNS information content management system

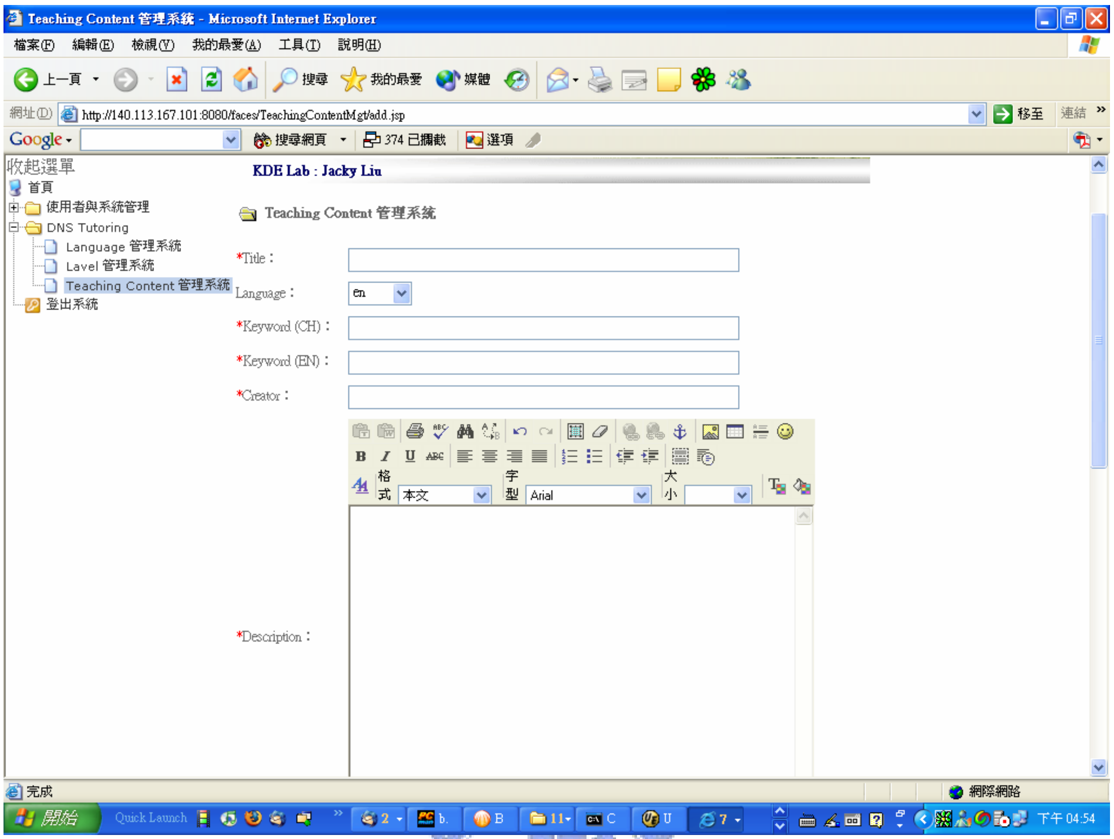
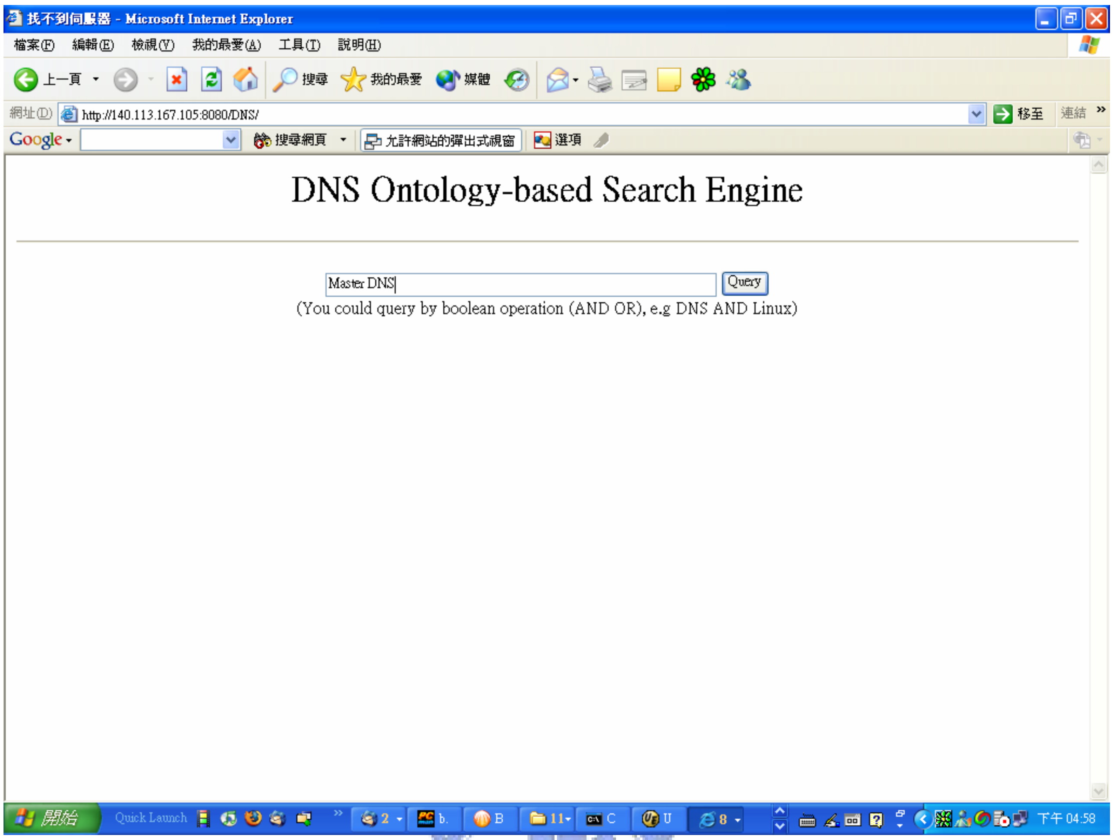


Fig. 7.8: DNS related data insertion interface

In our system, we would like to build the DNS knowledge portal. Therefore, we need a lot of DNS related information. In addition to the teaching materials, we gather the articles, FAQs and information in the books. The information would be located in different data sources. For example, some of the articles would be stored as HTML files in the file system and some of the articles are stored in the database. Furthermore, to facilitate the information gathering, we build a content management system (CMS) for the information management. Fig. 7.7 shows the CMS interface, which provides “add”, “delete” and “edit” functions on the data. Fig. 7.8 shows the data insertion interface and users could enter title, keyword, description, etc. information. The backend of CMS is database, so the information would be stored into the database. Hence, our system should take into account the data from different data sources. In addition, new data format may be imported in the future. Therefore, a flexible architecture which allows different data source without changing most of the design is required. As described above, we make use of ***Factory*** design patter to achieve the goal.

To speed up the search performance, index mechanism is required. ***Apache Lucene*** provides the index mechanism, so we could define what kind of data should be extracted for indexing. In addition, when the users enter the search query string, the search engine should transform the query string into semantic terms based on the DNS ontology. The whole inference process is described in Section 6.4. Fig. 7.9 shows the search interface where users could use keyword and ***Boolean*** operations**.** For example, if users would like to search information about ***DNS*** and ***Linux***, they could input the query string "***DNS AND Linux***". When the search engine receives the query string, the inference process would be active. As shown in Fig. 7.10, if the users input "***Master DNS***" query string, the inference engine would transform the query string into "***Master DNS OR Primary DNS***". Since the synonym relationship exists between ***Master DNS*** concept class and ***Primary DNS*** concept class. Therefore, the inference engine extend the original terms using ***OR*** operation on these two concept classes.

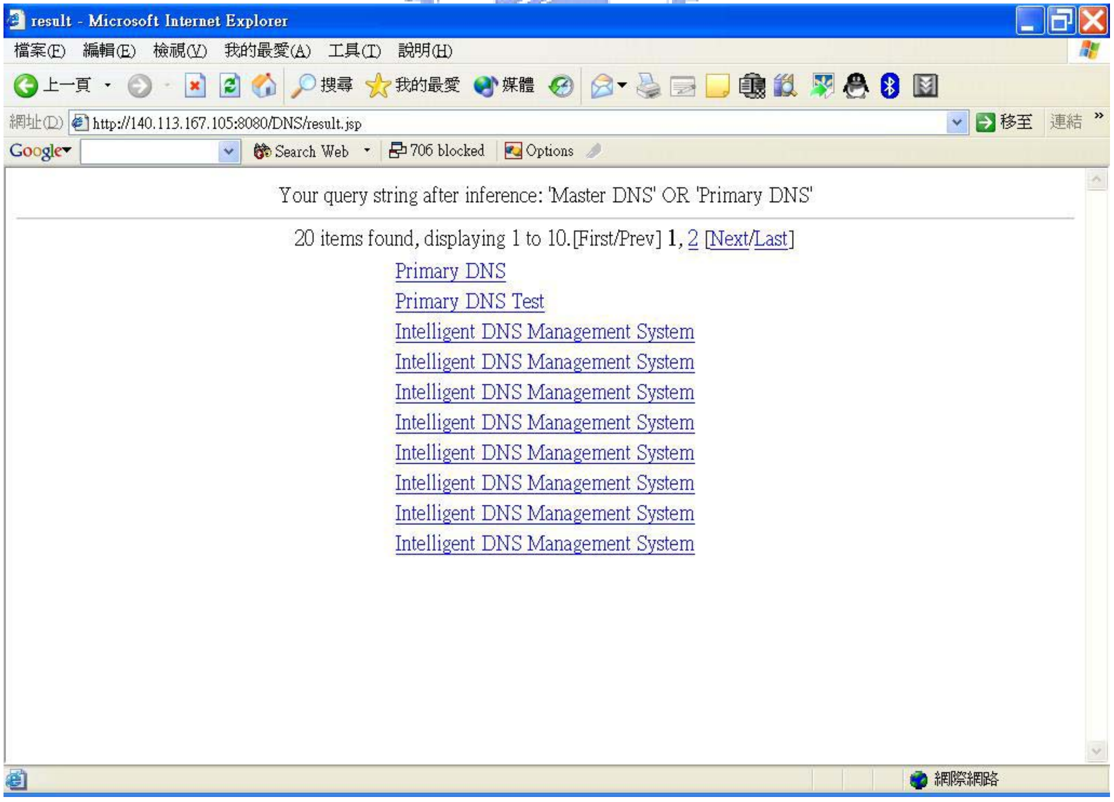

找不到伺服器 - Microsoft Internet Explorer
http://140.113.167.105:8080/DNS/
Query

DNS Ontology-based Search Engine

Master DNS

(You could query by boolean operation (AND OR), e.g DNS AND Linux)

Fig. 7.9: DNS ontology-based search interface


result - Microsoft Internet Explorer
http://140.113.167.105:8080/DNS/result.jsp

Your query string after inference: 'Master DNS' OR 'Primary DNS'

20 items found, displaying 1 to 10.[First/Prev] 1, 2 [Next/Last]

Primary DNS
Primary DNS Test
Intelligent DNS Management System
Intelligent DNS Management System
Intelligent DNS Management System
Intelligent DNS Management System
Intelligent DNS Management System
Intelligent DNS Management System
Intelligent DNS Management System
Intelligent DNS Management System

Fig. 7.10: Search result using query string “Master DNS”

7.6 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. 7.11 shows the daily statistics for March 2005 of our system. There are averagely 140 hits every day during September. In addition, we have a forum to collect the user feedbacks and bug reports.

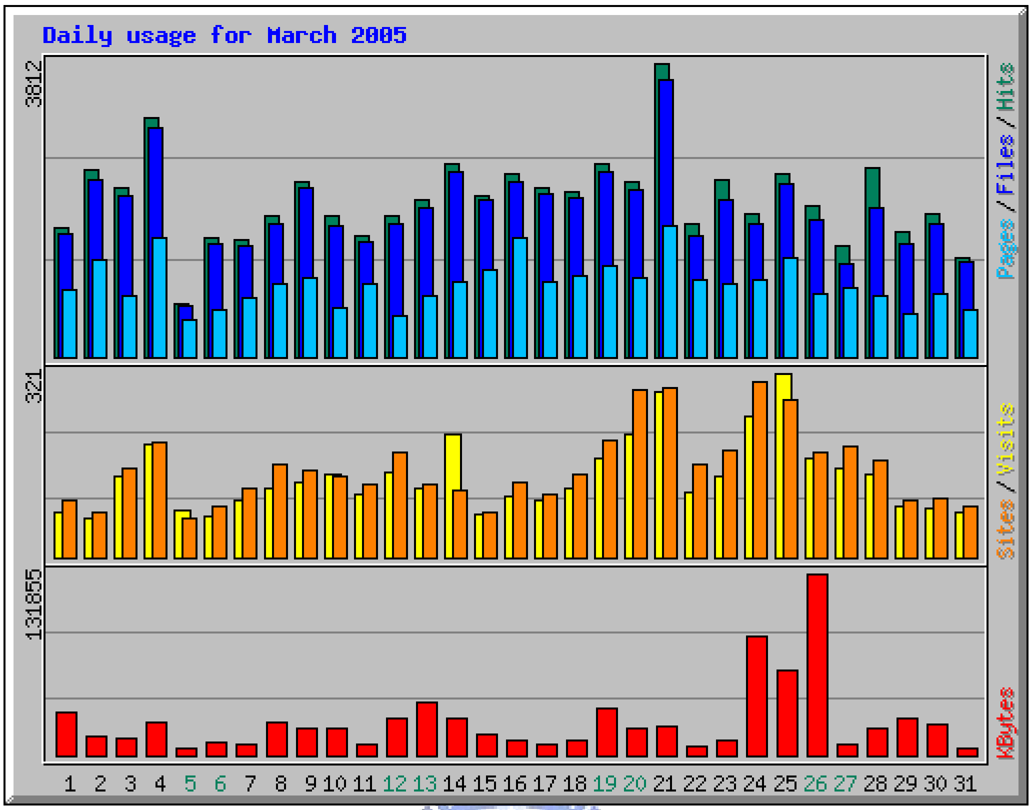


Fig. 7.11: Daily statistics for March 2005

Chapter 8 Concluding Remarks

In this thesis, we designed and implemented a DNS knowledge portal system. Our main contributions are: (1) to propose a DNS knowledge portal system (including DNS diagnosis service, DNS tutoring service and DNS ontology based search service) 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., (3) to propose an ontology-based model and algorithm for constructing the skeleton of model-tracing tutoring, which is used to work with the DNS diagnosis system to trace users' problems and activities, and (4) to propose a ontology-based search service framework, which could incorporate with ontology to enhance the capability of search service and the flexible design could be easily reused and extended.

The Domain Name System (DNS) is an essential part of the Internet infrastructure. However, few existing DNS professional web services could provide the DNS related knowledge. In addition, the new trend of DNS (such as IPv6 and ENUM) makes DNS management more complex. In Chen et al. (2003), a unifying intelligent system was proposed for DNS management, which provides the framework for DNS-related services. Although the diagnosis service could provide the suggestions, the suggestions information for some novice DNS administrators is not enough. In addition, for some people, if they would like to know the DNS operation model in more detail, the tutoring materials would be required. Moreover, DNS service is a sustained and evolving task, which means that both the human resources (e.g., the DNS administrators with good domain knowledge) and the system resources (e.g., the functionalities and protocols of the DNS software) of a site might need updating from time to time. In general, this will be a great challenging task. Therefore, DNS tutoring system which could provide the teaching materials after diagnosis service is required. Furthermore, to improve the reusability and interoperability issues of the teaching materials, we adopt SCORM (Sharable Content Object Reference Model) as web-based tutoring platform.

In this thesis, we use ***DRAMA/NORM*** as an expert system shell because of its client-server architecture and the object-oriented knowledge base structure. Based on the client-server architecture, it thus becomes very easy for us to develop KBS for supporting intelligent DNS management through www interface. On the other hand, 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, which facilitates the management of rules in each knowledge class. Second, since the ontology is mainly of an object-oriented structure. We can construct the object-oriented rule base more conveniently. Third, it is easy to reuse existing rules based on modular knowledge base design. Therefore, we can provide personalized service for different users.

According to the experimental results, the paradigm of using DNS ontology to facilitate constructing DNS model-tracing tutoring system works good and effective. The DNS tutoring system benefits the sharing and reusing of global DNS knowledge, the reduction of people's time to learn DNS management, and the improvement on the DNS and network operations. It is supposed that, with some minor adaptations, the same approach could be easily modified to many other engineering domains for facilitating knowledge base construction.

We have started to offer diagnosis service since 2003 and feedback shows that the paradigm of using DNS ontology to build knowledge-based system works good and effective. The integration of DNS diagnosis service, tutoring service and search service would benefits the sharing and reusing of DNS knowledge. In addition, with a few modifications, the same paradigm and developed algorithms could be easily adapted to other scientific or engineering domains. Future researches will focus on several issues. First, since the DNS system is still evolving, the DNS ontology should be evolved as well. Therefore, the new applications issues related to DNS (e.g., multilingual DNS, intrusion detection mechanisms concerning DNS, etc.) will be taken into account in the future. When the DNS ontology is more complete, DNS knowledge portal would cover more DNS related issues. Second, the extensions should be reflected on each of the appropriate services in our proposed DNS knowledge portal. Finally, new DNS related services should be incorporated into existing DNS knowledge portal to let more local DNS administrators gain more insight of DNS administration in a systematic and effective approach.

Reference

[AB+90] Anderson, J. R., Boyle, C. F., Corbett, A., & Lewis, M. (1990). Cognitive modeling and intelligent tutoring. Artificial Intelligence, 42, 7-49.

[AC93] Anderson, J.R. and Corbett, A.T. (1993). Tutoring of cognitive skill. In J.R. Anderson, Rules of the Mind (pp. 235-255). Hillsdale, NJ: Erlbaum

[AL01] Albitz, P. and Liu, C. (2001). DNS and BIND 4th edition, O'Reilly & Associates, Inc., Sebastopol, CA, 2001

[AP91] Anderson, J. R., & Pelletier, R. (1991). A development system for model-tracing tutors. In Proceedings of the International Conference of the Learning Sciences (pp. 1-8). Evanston, IL

[BC+01] Brownlee, N. , Claffy, k., and Nemeth, E., "DNS Measurements at a Root Server", Globecom 2001.

[Bellovin95] Bellovin, S.M., "Using the Domain Name System for System Break-ins", In Proceedings of Fifth Usenix UNIX Security Symposium, June 1995.

[BH+01] Berners-Lee Tim, Hendler James and Lassila Ora, The Semantic Web, "Scientific American, May 2001"

[BIND05] BIND (Berkeley Internet Domain), URL:http://www.isc.org, Accessed on Jan. 23, 2005.

[CERT00] CERT/CC. "CERT Advisory CA-2000-20 Multiple Denial-of-Service Problems in ISC BIND." 28 Nov. 2000 (revised). URL:

http://www.cert.org/advisories/ (9 Feb. 2001), etc.

[CH97] Callon, R., Haskin, D., "Routing Aspects of IPv6 Transition", IETF RFC 2185, September 1997.

[CJ+99] Chandrasekaran, B. and Jorn R. Josephson, V. Richard Benjamins. (1999). What Are Ontologies, and Why Do We Need Them? IEEE Intelligent Systems. 14 (1): pp. 20 - 26.

[Cockburn97] Cockburn, Alistair. (1997). *Structuring Use Cases with Goal.* Journal of Object-Oriented Programming, in two parts, the Sep-Oct issue and the Nov-Dec issue.

[CT+02-1] Chen, C.S., Tseng, S.S., Liu, C.L. (2002). A distributed intrusion detection model for the domain name system. Special Issue on Parallel and Distributed Systems, Journal of Information Science and Engineering, Vol.18, pp.999-1009.

[CT+02-2] Chen, C.S., Tseng, S.S., Liu, C.L., Ou, C.H. (2002). Building a DNS ontology using METHONTOLOGY and Protege-2000. In Proceedings of 2002 International Computer Symposium Workshop on Artificial Intelligence, Dec. 18-21, 2002 .

[CT+03] Chen, C.S., Tseng, S.S., Liu, C.L. (2003). A unifying framework for intelligent DNS management. *International Journal of Human - Computer Studies*, Vol. 58/4, pp 415 – 445.

[DNSBL03] DNSBL (DNS Blocking List), How ip4r (DNSBL-style) DNS lookups work, http://www.declude.com/JunkMail/Support/ip4rinfo.htm, 2003

[DNSreport05] DNSreport, http://www.dnsreport.com, Access on Jan 25, 2005.

[DS+93] Davis Randall, Shrobe Howard, Szolovits Peter, "What is a Knowledge

Representation?", *AI Magazine*, 14(1):17-33, 1993

[Durkin94] Durkin, J. (1994). Expert System: Design and Development, Macmillan Publishing Company, 1994.

[Faltstrom00] Faltstrom, P.,"E.164 and DNS", IETF RFC 2916, September 2000.

[Faltstrom03] Faltstrom, P., Hoffman, P., Costello, A., "Internationalizing Domain Names in Applications (IDNA)", IETF RFC 3490, March 2003

[Fernandez99] 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. CEUR Publications.

[FG+97] 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.

[Gaines00] Gaines, B.R., Knowledge Science and Technology: Operationalizing the Enlightenment. In Proceedings of PAKW2000

[Gennari+03] Gennari, J.H., et al., The Evolution of Protégé: An Environment for Knowledge-Based Systems Development, *International Journal of Human-Computer Interaction*, 58(1), pp. 89-123, 2003

[Gruber93] Gruber, T. R., "A translation approach to portable ontologies", *Knowledge Acquisition*, 5(2):pp.199-220 (1993).

[GS92] Gaines, B.R., and Shaw, M.L.G. (1992). *Knowledge Acquisition Tools based on Personal Construct Psychology*. Special Issues on "Automated Knowledge Acquisition Tools" of the Knowledge Engineering Review.

[GS93] Gaines, B.R., and Shaw, M.L.G. (1993). *Eliciting Knowledge and Transferring it Effectively to a Knowledge-Based System*, IEEE Transactions on Data and Knowledge Engineering, 5(1), pp.4-14.

[Hanley00] Hanley, Sinead. "DNS Overview with a discussion of DNS Spoofing." 6 Nov. 2000. URL: http://www.sans.org/infosecFAQ/DNS/DNS.htm (9 Feb. 2001).

[HD98] Hinden, R., Deering, S.,"IP Version 6 Addressing Architecture", IETF RFC 2373, July 1998

[HH02] Haddad, I. and Hawwa, S., "The evolution of networking protocols to meet the requirements of 3G services", Tutorial Program of ACM Multimedia 2002, December 1-6, 2002

[HS+97] Heijst, G.V., Schreiber, A.T., and Wielinga, B.J.. (1997). Using Explicit Ontologies in KBS Development, *International Journal of Human-Computer Studies, Vol. 46, No. 2/3, pp. 183-292.*

[HS+99] Handley, M., Schulzrinne, H., Schooler, E., Rosenberg, J., "SIP: Session Initiation Protocol", IETF RFC 2543, March 1999.

[Kobryn99] Kobryn, Cris. (1999). UML 2001: A Standardization Odyssey. In Communications of the ACM, vol. 42, no. 10.

[Koh01] 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.

[KP98] Krasner, G.E. and Pope, S.T. (1988.) *A Cookbook for Using the Model-View-Controller User Interface Paradigm in Smalltalk*-80. Journal of Object-Oriented Programming, 1(3), 26-49.

[LT+03] Lin, Y. T., Tseng, S. S., Tsai, C. F. (2003). "Design and Implementation of New Object-Oriented Rule Base Management System", Journal of Expert Systems with Applications, Vol. 25, pp369-385.

[LT+03-1] Liu, C.L., Tseng, S.S., Chen, C.S., (2003). "Email Attack Ontology for SPAM Fighting", in Proceedings of ICACT2003 (5th International Conference on Advanced Communication Technology), Jan. 20-22, 2003, Korea.

[LT+03-2] Liu, C.L., Tseng, S.S., Chen, C.S. (2003). "The Design of Anti-SPAM Knowledge-Based System", in Proceedings of Apricot2003, Taiwan

[LT+04-1] Liu, C.L., Tseng, S. S. and Chen, C. S. "Design and Implementation of an Intelligent DNS Management System", Expert Systems with Applications, Volume 27, Issue 2, August 2004, pp. 223-236

[LT+04-2]Liu, C.L., Tseng, S. S. and Chen, C. S. "Design of an ENUM DNS Server on a Hybrid IPv6/IPv4 Internetworking Environment", in Journal of Internet Technology, Vol.5 No.2 (2004)

[LT+04-3]Liu, C.L., Tseng, S. S. and Chen, C. S. "Ontology-based DNS Model-Tracing Tutoring System", Submitted to *International Journal of Human - Computer Studies 2004*

[M&M03] Man-Mice Company. (2003). Domain Health Survey for .COM - February 2003, http://www.menandmice.com/6000/61_recent_survey.html

[MD00] Mealling, M. and. Daniel, R., "The Naming Authority Pointer (NAPTR) DNS Resource Record", IETF RFC 2915, September 2000.

[Mockapetris87-1] Mockapetris, P. (1987). "Domain Names - Concepts and Facilities," RFCs 1034, November 1987.

[Mockapetris87-2] Mockapetris, P. (1987). ``Domain Names - Implementation and Specification" *RFC 1035*, Nov. 1987

[NF+00] Noy, N. F., Fergerson, R.W., Musen, M.A. (2000). The knowledge model of Protege-2000: Combining interoperability and flexibility. Proceeding of the 2th

International Conference on Knowledge Engineering and Knowledge Management (EKAW'2000), Juan-les-Pins, France.

[NS+00] Nemeth, E., Snyder, G., Seebass, S., Hein, T.R. (2002), **UNIX System Administration Handbook (3rd Edition)**, Prentice Hall PTR; August 2000.

[Shirky00] Shirky, Clay. "*What is P2P... And What Isn't?"* The O'Reilly Network, 24 Nov 2000

[SO+00] Shadbolt, N., O'Hara, K., and Cottam, H. (2000). The Use of Ontologies for Knowledge Acquisition. In: J. Cuena, et al., (eds) Knowledge Engineering and Agent Technology. IOS Press, Amsterdam.

[Wu00] Wu, X., "Knowledge object modeling," *IEEE Transactions on System, Man, and Cybernetics*—Part a: Systems and Humans, Vol.30, NO.2, March 2000