DNS Ontology-based Search Service

Chieng-Liang Liu(劉建良)★ Chang-Sheng Chen(陳昌盛)‡ Shian-Shyong Tseng(曾憲雄) ★
Department of Computer & Information Science★,
Computer & Network Center‡
National Chiao Tung University, Hsinchu, Taiwan
Corresponding E-mail: cschen@mail.nctu.edu.tw

摘要

本文主要研究內容，在於設計並實作完成一套以DNS 本體為基礎的搜尋服務，由原先的『診斷-教學』模式，進一步再提昇成為『診斷-搜尋-教學』相戶結合的新模式，以強化原有的DNS 知識入口網站功能 (http://idns-kde.nctu.edu.tw)，以利用戶透過線上學習 DNS 相關的管理知識 。

關鍵詞：網域名稱系統，知識入口網站，知識分享，知識本體，語意搜尋服務。

Abstract

In this paper, we design and implement a DNS ontology-based, three-layer (i.e., presentation layer, logic layer and data layer) search service for enhancing the semantic search to strengthen DNS knowledge sharing through the DNS knowledge portal, iDNS-KDE [7], to facilitate people in locating the required information and help alleviating the system management issues on the DNS domain.

Keywords: DNS, knowledge portal, knowledge sharing, ontology, semantic search.

1. Introduction

As shown in Man-mice [3], nearly 70% of the Domain Name System (DNS) [1] servers of commercial sites (e.g., ".COM" Zones) have some configuration errors. In essence, the problem is mainly due to the distributed nature of DNS systems and lack of efficient knowledge sharing mechanisms among DNS administrators. In practice, online tutoring system (or something of the like), if designed and implemented properly, could provide DNS-related knowledge during DNS construction and maintenance phases to help deal with the above issues and reduce the percentage of mis-configuration.

† This work was partially supported by MOE and National Science Council of the Republic of China under Grant No. NSC94-2752-E-009-006-PAE, Grant No. NSC93-2213-E-009-099 and Grant No. NSC94-2213-E-009-111.

‡ Corresponding author.

In [3], we proposed a unifying intelligent system framework (i.e., iDNS-KDE [7]) for DNS management, including DNS configuration, DNS design, DNS diagnosis, and DNS tutoring systems. In [10], we further enhanced the online diagnosis of iDNS-KDE. However, from users' feedbacks, we found that even though the diagnosis service could provide the suggestions, the suggestive information was not helpful enough for some novice DNS administrators. Sometimes, they would like to know more details about the DNS operation model and related information. Therefore, the tutoring materials would be required.

But, there comes another big problem. It is usually hard for these novice people to find the right information needed since there is so much information in the Internet. Therefore, we need to add search services (and extend the *diagnosis-and-tutoring* model to become "*diagnosis-search-tutoring*") to facilitate people in locating the required information. In practice, most traditional search services are based on keyword matching processes without considering the semantic information embedded in the search strings, which may often lead to inappropriate results. The drawbacks of keyword search are briefly described as follows:

(1). ***Ambiguity problems***: Term ambiguity often occurs during keyword search and leads to un-related information result. For example, the single term "spoofing", refers especially to altering TCP/IP packet source addresses or other packet-header data in order to masquerade as a trusted machine (e.g., by capturing, altering, and retransmiting a communication stream in a way that misleads the recipient). However, while dealing with DNS problems, there are two common spoofing cases that might confuse novice DNS administers (or users). That is,

- *IP spoofing* (IP-layer issue): IP spoofing is the creation of IP packets using someone else's IP address.
- *DNS spoofing* (Application-layer issue): DNS spoofing is the substitution of a different IP address for a DNS name.

In both instances, the computer you are connecting to is not the server you expect. But, these two kinds of problems are caused by different mechanisms and we should take actions in different places to avoid them. For example, DNS spoofing is commonly achieved by corrupting the DNS database of the DNS server (your computer connects to) in order to match human readable computer names to physical IP addresses. Without any other information providing, the search engine might misunderstand the meaning and return inappropriate results to the query user.

(2). ***Expression problems*:** Sometimes, it is not easy for people to express what really they want with keyword expressions. For example, many novice DNS administrators might know that the general term "DNS security" could be used to indicate DNS-related security issues, but they might not know enough on how to differentiate among other more specific DNS security issues (e.g., "DNS Spoofing", "DNS Dynamic Update" or "Zone Data Protection", etc.) when they encounter problems.

(3). ***Synonym problem*:** Sometimes there might be different terms that refer to the same things. For example, the following two sentences are treated as the same in the DNS.

- The DNS server X is the "**master DNS server** of the domain zone Y".
- The DNS server X is the "**primary DNS server**" of the domain zone Y.

In the above two sentences, both "master DNS server" and "primary DNS server" are synonyms in the DNS. Without the background knowledge, the users would miss some required information. When users would like to search "master DNS server" information, "primary DNS server" information may be excluded.

In this paper, we propose a DNS ontology-based, three-layer (i.e., presentation layer, logic layer and data layer) search system framework for enhancing the semantic search to strengthen the DNS knowledge sharing mechanism, with the design and implementation of the DNS knowledge portal, iDNS-KDE [7]. On the one hand, as mentioned in [2, 4, 5, 6], ontology could represent the semantics of the terms. Therefore, we adopt the ontology as the semantics resolution mechanism to improve the search capability. On the other hand, the separation of the layers could make the whole system more flexible and reusable.

In the following, we will briefly describe the organization of the rest of the paper. Section 2 gives overviews of the background and related issues for highlighting the directions of the research. Information in Section 3 describes about the system architecture and the design principles. Section 4 contains some implementation details and discussions issues. Finally, in Section 5, a concluding remark and some points are highlighted for future research.

2. Background and Related Work

2.1 Online diagnostic services of iDNS-KDE

Even though DNS is so important to network operation, few DNS administrators have the expertise to do the jobs well. Improperly configured (or deployed) DNS traffic or intent anomalous activities (i.e., DNS domain zone scanning, SPAM mails, intrusion attempts, etc.) from local and remote sites might greatly affect the operation of local DNS systems and the related network from time to time. The phenomenon is especially obvious between novice and inexperienced administrators that manage a small scale of network.

In [3], we proposed a unifying intelligent system framework (i.e., iDNS-KDE[7]) for DNS management, including DNS configuration, DNS design, DNS diagnosis, and DNS tutoring systems.). ***iDNS-KDE***, developed by using web interface and expert system technology, began to offer public access services since May 2003. In [10], we further enhanced the online diagnosis of iDNS-KDE. On the other hand, many feedbacks on iDNS-KDE also show that a lot of novice users would require more details and guidance about the design principle (e.g., many users do not fully understand the obtained DNS diagnosis results, due to their lack of theoretical and practical knowledge of DNS system), but did not have the clues on where to start acquiring the related knowledge or required information.

2.2 Traditional Search System

Most traditional search systems compute the similarities between objects (or concepts) based on the term frequency (TF) or inverse document frequency (IDF). However, in practice, if we consider only the terms, we would often miss the semantic information of the terms because it is difficult for most people to describe the terms correctly. For example, there are many DNS security-related issues (e.g., DNS spoofing, DNS zone data protection, etc.). Usually, for most users, what they could describe is only the term "DNS security". In other words, the general terms expression is easy for most users. Especially, when we are not familiar with the domains, this becomes obvious.

Next, there are synonym issues for most domains. If we consider only keyword matching, the synonyms of the query string will be ignored and 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.

2.3 Ontology as semantic guide

Ontologies are useful in a range of applications, where they provide a source of precisely defined terms for communicating among 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.

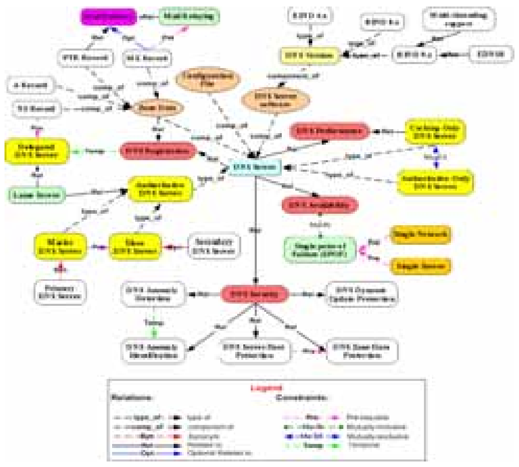

Fig.1: Skeleton of a DNS ontology

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 [5]. With the help of ontology, the knowledge is not only human-readable but also machine-readable. Fig. 1 shows the skeleton of a DNS ontology as proposed in [3].

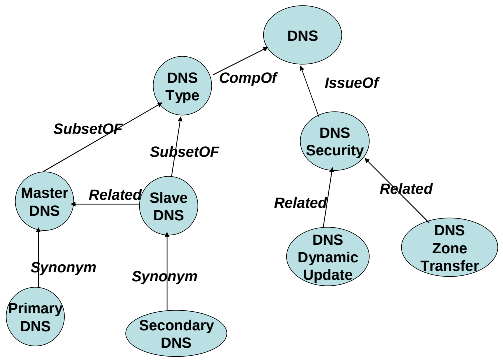

Fig.2: Partial DNS ontology

As shown in Fig. 2, the partial DNS ontology could represent the relationship between DNS concepts. Generally speaking, we could represent semantic information by the attributes of ontology concepts 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 with much background domain knowledge. First, the taxonomy hierarchy information could provide us with the inheritance information. As shown in Fig. 2, the "*SubsetOf*" relationships 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, 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. From the ontology, we could know that, 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 there is a "*SubsetOf*" relationship between concept ***A*** and concept ***B,*** and another "*SubsetOF*" relationship between concept ***B*** and concept ***C***, we would infer that there is a "*SubsetOF*" relationship between concept ***A*** and concept ***C***.

3. DNS Ontology-based Search Service

In this paper, we propose an ontology-based search service for enhancing the semantic search to strengthen DNS knowledge sharing with the design and implementation of a DNS knowledge portal in web services.

As shown in Fig. 3, the whole process could be described as follows: (1) A user submits a query string to the search service system; (2) The search service engine 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.

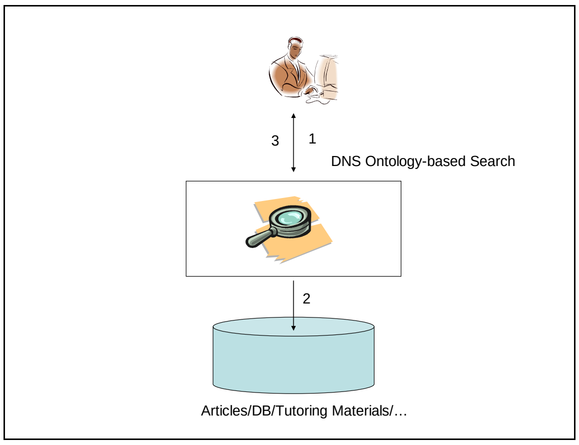

Fig. 3: DNS Ontology-based Search Service

3.1 System Architecture

Fig. 4 shows the overall system architecture of the DNS ontology-based search engine. As mentioned in Sec. 2.2, ontology could provide the taxonomy hierarchy information to provide basic inference mechanism for information reasoning and improve the capability of search system. Different ontology representations might 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. The input is the query string from users. The inference engine could support many ontology representation formats when we provide the required implementations. The advantage of the design is to separate the interface and the implementations to improve the reusability of the system.

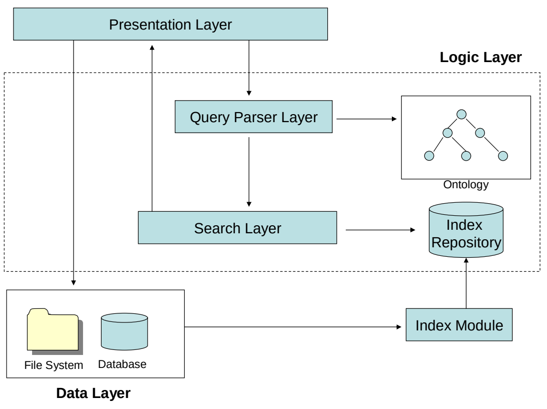

Fig. 4: System architecture

Furthermore, a scalable system which could provide robust services is an important design goal. Therefore, in the system design, we take into account these issues and propose a three-layer framework. As shown in Fig. 4, the whole system could be divided into the presentation layer, the logic layer and the 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 the logic layer is to act as the bridge between the presentation layer and the data layer. It could be further divided into two sub-layers, query-parser layer and search layer. The logic layer will receive query inputs from the presentation layer and the query-parser layer will trigger the internal inference engine based on DNS ontology. The inference result will then be passed 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.). Furthermore, to speed up the search process, it is necessary to index these data sources. 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 archives, the webpages in the Internet, news, blog, etc.) are imported, the data layer should be able to handle the new data source without changing the existing design.

3.2 MVC design pattern

Just like the *Model-View-Controller* (MVC) design pattern [**8**], our design focuses on the separation of the presentation, the logic processing and the data layers for facilitating the flexibility of the system design. That is, if there is a requirement to change the design of any layer, we do not have to change the design of the other layers. For example, if we add more data sources into the data layer, the logic layer could still access the result of the index and the presentation layer presents the result as usual. Furthermore, to enhance the ontology reasoning process, the ontology inference engine is abstractly designed since the ontology representation we choose might vary, depending on different application environments. For example, we could represent the ontology by using XML, RDF, or OWL, etc.

Fig. 5 shows the ontology inference engine design flow. The inputs are query strings from users. The inference engine could support many ontology representation formats when we provide the required implementations. The advantage of the design is separate the interface and the implementations and that could improve the reusability of the system.

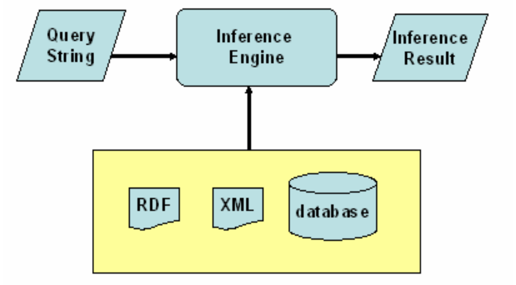

Fig. 5: Ontology Inference Engine Flow

3.3 The Search Engine

In our system design, we make use of the ***Lucene [9]*** query parser mechanisms to represent the final inference result. Furthermore, we adopt XML as the ontology representation format.

Fig. 6 shows part of the DNS ontology XML. 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 Slave DNS***)".

```
<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>

    …


</Ontology>
```

Fig. 6: DNS Ontology XML

In addition, we define the "*Related*" relationship. The related relationship could be used to model general terms condition. For example, if the users would like to find out the documents about DNS security issues. Since DNS security consists of many related issues (e.g., ***DNS Spoofing***, ***Zone Data Protection,*** etc.), most users do not know the details on these issues. In most of the traditional search systems, the users may miss some information by using keywords matching only. Therefore, in our system, we define the ***Related*** relationship to solve this kind of problem.

4. Implementation and Discussion

4.1 Implementation

As mentioned, we build a DNS knowledge portal (i.e., **iDNS-KDE)**, which needs a lot of DNS knowledge sources, for facilitating DNS knowledge sharing (or learning) among DNS administrators. In addition to the teaching materials from domain experts, we gather articles, FAQs and information from many different sources (e.g., DNS-related books and Internet sites). For example, some articles are stored as HTML files in the file system and other articles might be stored in the database. Furthermore, to facilitate the information gathering, we build a simple content management system (CMS), which provides "add", "delete" and "edit" functions on the data, for the information management. Fig.7 shows the data insertion interface and users could enter title, keyword, or description etc. information.

Next, to speed up the search performance, an indexing mechanism is required. ***Apache Lucene*** provides the index mechanism, and 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 (or expand) the query string into semantic terms based on the DNS ontology. The whole inference process is described in Sec. 3.3.

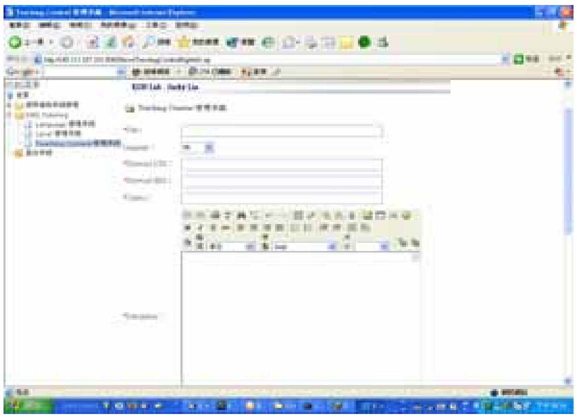

Fig. 7: A Content-Management System

When the search engine receives the query string from a user, the inference process would be active. For example, as shown in Fig. 8, after the user 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 extends the original terms using ***OR*** operation on these two concept classes.

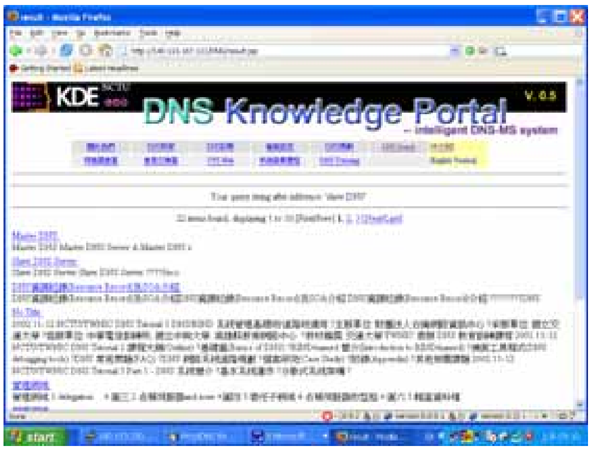

Fig. 8: Search Result for the Query "Master DNS"

4.2 Lucene [9] – search engine library

Apache Lucene (http://lucene.apache.org/) is a high-performance, scalable, full-featured text search engine library written entirely in Java. The powerful abstractions and useful concrete implementations make *Lucene* very flexible. Lucene provides basic index and search architectures and it is easy to extend Lucene for more advanced indexing or search.

Lucene only accepts plain text format, therefore, we have to transform the raw data (HTML files, database records, plain text etc.) into plain text format first and then break the plain text files into *Field* objects (name-value pairs). On the other hand, at the heart of Lucene is an index. Lucene searcher searches the index files for information retrieving. The basic element of Lucene index is *Field* object, which is a name-value pair. For example, a document may contain title and content fields:

- ***name:value***
- *title:DNS Availability Issies*
- *content: DNS SPOF problem is one of the DNS availability issues*

As for the search section, *Lucene* provides the query parser mechanisms and they could fulfill most of your requirements. For example, most of the search engine provides 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 also 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.

4.3 Discussions

In this paper, we propose a DNS ontology-based, three-layer (i.e., presentation layer, logic layer and data layer) search system framework. The separation of the layers could make the whole system more flexible and reusable. For example, different ontology representations (e.g., XML, RDF, PWL, etc.) might 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 for supporting many ontology representation formats (i.e., if there is a requirement to adapt to new ontology sources). The advantage of the design is to separate the interface and the implementations to improve the reusability of the system when we start the required implementations. With some minor modifications, we could change the presentation from web interface to other user interfaces (e.g., PDA, mobile phone, etc.). In addition, the flexibility of importing new data source is also reserved. For example, when different data sources (e.g., the mailing list archives, PDF files, WORD files, the webpages in the Internet, news, blog, etc.) are imported, the data layer is able to handle the new data source without changing the existing design.

5. Concluding Remarks

In this paper, we design and implement an ontology-based search service for enhancing the semantic search to strengthen DNS knowledge sharing through the DNS knowledge portal, iDNS-KDE, to facilitate people in locating the required information and help alleviating the system management issues on the DNS domain. In our design, we propose a three-layer (i.e., presentation layer, logic layer and data layer) DNS ontology-based search system framework. The separation of the layers could make the whole system more flexible and reusable.

Future researches will focus on several issues. First, since the DNS system is still evolving, we will extend the data sources of the ***iDNS-KDE*** (and the related DNS ontology) to integrate new topics such as IPv6 and multi-lingual issues concerning DNS. Second, there might be a need to incorporate new and different data sources (e.g., the mailing list archives, PDF files, WORD files, the webpages in the Internet, news, blog, etc.). The data layer should be adjusted to handle the new data source without changing the existing design. Furthermore, with some minor modifications, we could change the presentation from web interface to other user interfaces (e.g., PDA, mobile phone, etc.) to improve the knowledge sharing and reusability of the system.

References

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

[2] 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.*

[3] 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.

[4] 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.*

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

[6] 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.*

[7] iDNS-KDE system, http://idns-kde.nctu.edu.tw

[8] 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.*

[9] **Lucene,** http://lucene.apache.org/Java/docs/

[10] Liu, C.L; Tseng, S.S.; and Chen, C.S.; "Design and Implementation of an Intelligent DNS Management System", *Journal of Expert Systems With Applications, September 2004*