Facilitating KMS Reusability by XML Binding Model[*]

Sheng-Tun Li
Institute of Information Management
National Cheng Kung University
No.1, Ta-Hsueh Road, Tainan 701, Taiwan
stli@mail.ncku.edu.tw

Huang-Chih Hsieh
Department of Information Management, National Kaohsiung First University of Science Technology
2 Juoyue Rd. Nantz District, Kaohsiung 811, Taiwan
jack@ai.nkfust.edu.tw

Shu-Ching Chen
Distributed Multimedia Information System Laboratory
School of Computer Science
Florida International University
Miami, FL 33199, USA
chens@cs.fiu.edu

Mei-Ling Shyu
Department of Electrical & Computer Engineering
University of Miami
Coral Gables, FL 33124, USA
shyu@miami.edu

Abstract - *Since knowledge management becomes an important issue, knowledge management system (KMS) plays a crucial role to enhance the performance of knowledge management. It is a difficult work to build a KMS from scratch. In order to facilitate KMS reusability, we propose a scheme based on XML binding technology which can automatically generate the knowledge management components so that one may reengineer the system to accommodate with different requirements of other domains.*

Keywords: Knowledge Management System, reusability, XML Binding Model, JAXB

1 Introduction

Knowledge management (KM) is a complex problem and is related to many issues, such as socio-organizational, financial, economical, technical, human, and legal concerns [1]. The basic activities of KM include identification, acquisition, development, dissemination, use, and preservation of the enterprise's knowledge [2]. No matter what process is, the objective of KM is to promote knowledge growth, knowledge communication and knowledge preservation in an organization [3]. In order to achieve this goal effectively and efficiently, the information technology has been considered as an active enabler of KM [4], and there exist different KMS in facilitating the activities of KM [5][6][7].

Since knowledge is treated as an important asset of enterprise, knowledge management system (KMS) becomes a crucial enabler to facilitate the activity of knowledge management. To build a complete knowledge management system is a difficult and arduous task, and it is very time consuming. The question arises whether we can construct a new KMS that is based on existing KMS, or to apply existing KMS to other industries. However, it is not an easy task because of the different characteristics of domain knowledge.

Before we apply existing KMS to other industries, we have to understand the type of KMSs. According to [8], KMS can be classified into two models: Model 1 KMS is for routine and structured information processing; Model 2 KMS is for non-routine and unstructured sense making. The goal of Model 1 KMS is "getting the right information to the right person at right time". In short, model 1 KMS is based on doing the thing right. In model 2, the KMS can be characterized as intelligence in action, such as attention, motivation, commitment, creativity, and innovation. Usually, model 1 KMS reusability is higher than model 2 KMS due to routine and structured information processing. Although model 1 KMS has high reusable, system developers still need to spend many efforts to redesign the existing KMS parts in order to fit into the new KMS for other industries. In this study, we developed a KMS that belongs to model 1 KMS, and we proposed a XML binding technology scheme which can automatically generate the knowledge management components so that one may reengineer the system to accommodate with different requirements of other domains.

The remaining parts of this paper are organized as follows. Section 2 gives an overview of developed system. The XML binding technology is briefly introduced in section 3. Section 4 describes the XML binding model in this study. The system demonstration is sown in section 5. Section 6 makes a conclusion.

2 System Overview

A knowledge entity can be treated as a knowledge object (KO). KOs can be numerical data, text streams, validated models, meta-models, movie clips, or animation sequences [9]. Since enterprises are interested in the integration of existed knowledge bases [10], how to integrate and share KOs among different KMS is in great demand and is a challenge task. In the literature, metadata has been widely used in the integration of existing knowledge bases [11] whereas the ontology has been considered as a meta-level description of knowledge presentation [12].

[*] **© 2003 IEEE**

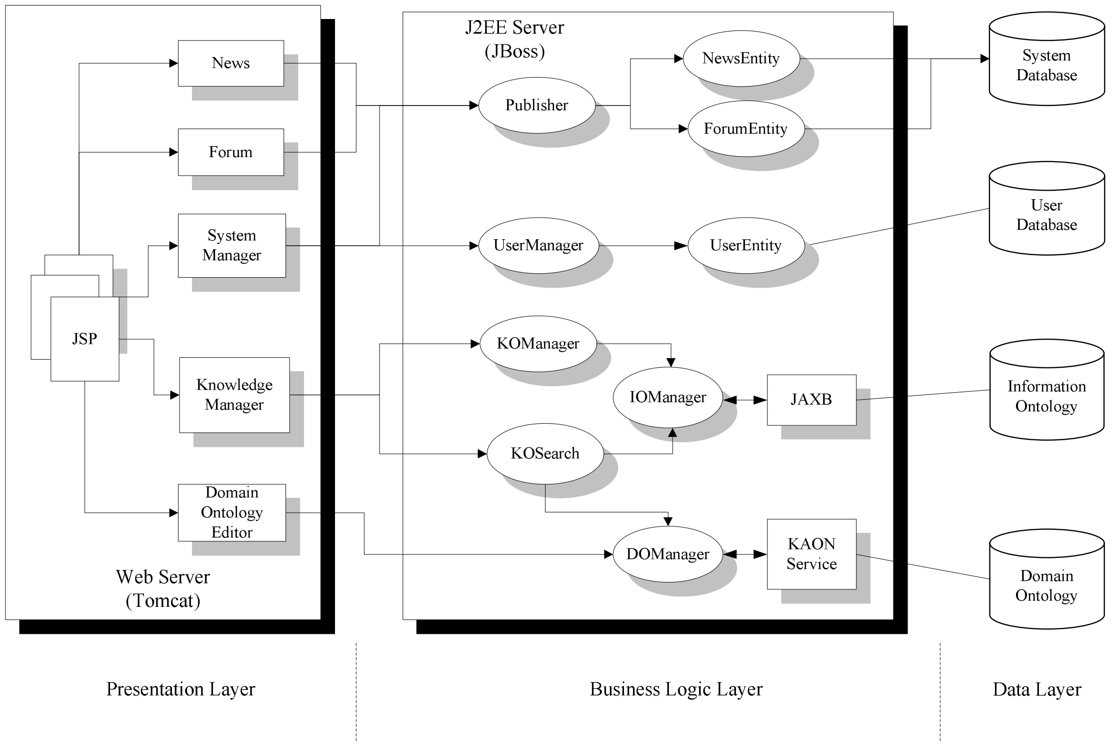


Figure 1. The architecture of knowledge management system

In this study, the goal of the proposed KMS is to assist users in sharing, searching, and managing knowledge objects. We make use of ontology to enhance the performance of knowledge search. In our study, ontology is classified into the information ontology and the domain ontology so that the semantic match of knowledge search can be realized. The information ontology is a meta model that describes knowledge objects and contains generic concepts and attributes of all information about knowledge objects, such as the title, authors, date, keywords, and other related information. The domain ontology consists of the concepts, attributes and instances of one industry. The purpose of domain ontology is to achieve the objective of semantic match when searching for knowledge objects.

The architecture of proposed system is shown in figure 1, which is composed of three layers: Presentation Layer, Business Logic Layer, and Data Layer. Due to space limitation, only the kernel components and the design philosophy of the system will be discussed.

KOManager component is a Java session bean that can create, share, browse, and remove knowledge objects. Once knowledge objects has been altered, information ontology also has been modified by IOManager that is based on JAXB (Java Architecture for XML Binding) [13] technology, to be discussed in next section. For knowledge searching, the KOSearch component provides an ontology-based search engine, which can search the domain and information ontology base through the DOManager and IOManager components, JAXB, and the KAON Service [14]. The KAON Service provides APIs to access domain ontology base. The search approach of the KOSearch component provides two models: the information-ontology that searches knowledge objects by keyword exact-matching and the domain-ontology that expands the keyword by the domain ontology.

The proposed system is developed under a cooperation project of industry and academia. The objective of this project is to introduce knowledge management to Metal Industries Research & Development Centre (MIRDC), a non-profit organization established in October 1963 for researching and developing the leading technology of metal and its related industries in Taiwan. When we apply this system to other industries, the system developers have to redesign the KO management components and information ontology. It is a tedious work. Thus, we proposed a scheme based on XML binding technology to facilitate the KMS reusability.

3 XML Binding Technology

There are two standard binding solutions of XML, namely, the Simple API for XML (SAX) and the Document Object Model (DOM) API. The parsing approach of SAX is even-driven, and it does not store data in memory so that it is appropriate for high-speed processing of XML. DOM, on the other hand, allows an application to manipulate the content in memory because it produces a presentation of the data in memory. However, both SAX and DOM programmer must provide necessary extra code to handle XML data, it might be complicated, error-prone, and difficult to maintain. If we could simply bind an XML document to an object, it would be much easier to write XML-enabled programs, and the programmer does not take care of how to parse XML data anymore. The vision is realized by Java Architecture for XML Binding (JAXB).

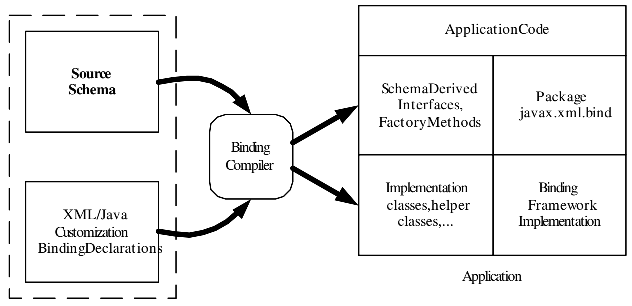

Source Schema
XML/Java Customization BindingDeclarations
Binding Compiler
ApplicationCode
SchemaDerived Interfaces, FactoryMethods
Package javax.xml.bind
Implementation classes,helper classes,...
Binding Framework Implementation
Application

Figure 2. JAXB architecture overview [15]

JAXB is released by Sun Microsystems Inc, which provides java developers an efficient and convenient way of mapping between XML and Java code. Figure 2 shows the architecture of JAXB. In the left hand side, The Source Schema describes the relationships among elements, attributes and entities in XML document. The binding declarations are rules to generate a set of Java codes. When the default binding rules are insufficient for developers need, JAXB supports customizations and overrides to default binding rules. The core of JAXB architecture is the Binding Compiler, which could transform or bind a source XML schema to a set of JAXB content classes. Finally, schema-derived classes and interfaces with necessary Java packages are generated by JAXB binding compiler in the right hand side.

4 XML Binding Model for Knowledge Objects

In this study, we define the information ontology as the knowledge object meta-data that makes knowledge objects easy to be understood and read. The most popular markup language of metadata is XML, and it is an industry-standard and system-independent way to present data. Thus it can be seen, XML is appropriate for implementing I nformation ontology, and it can facilitate the exchange of knowledge objects between KMSs.

Since Sun Microsystems Inc. releases JAXB, dealing with XML documents becomes an easy work. JAXB provides a fast, convenient way to create two-way mapping between XML documents and Java objects. The developer gives a XML schema which specifies the structure of XML data, the JAXB compiler could generate a set of Java classes according to the specified XML schema. Then, the developer does not need to take care of complex parsing and processing code any more. In this study, we make use of JAXB to manipulate information ontology. Although JAXB provides a convenient way to bind XML data to Java objects and the KMS developer could easily deal with XML data, the KMS developer still need to spend many efforts to rewrite KO management component when applying existing KMS to other industries. In order to facilitate the reusability of KMS, we also propose a scheme based on XML/Java technologies which can automatically generate the KO management (KOM) components so that one may reengineer the system to accommodate with different requirements of other domains.

The following subsections give details of the proposed scheme. The XML schema of knowledge object is described in section 4.1. Section 4.2 shows the framework of XML binding model. The proposed scheme of automatic component generation is explained in section 4.3.

4.1 XML Schema of Knowledge Object Base

As mentioned previously, information ontology is meta-data of knowledge objects, and we will bind information ontology to Java objects by using JAXB. At first, the system developer specifies the XML schema of knowledge object (KO) base. The parts of XML schema this study specifies is shown in Figure 3. As Figure 3 indicates, a KOBase element contains the Usage element whose type is string, the Owner element whose type is also string, and the unbounded KO element whose type is KnowledgeObject. The KnowledgeObject type, here, is the so-called information ontology which contains fifteen elements to describe the information of knowledge objects. Following, the JAXB compiler can compile this XML schema and generate Java classes that bind to information ontology. Finally, the system developer can easy manipulate and maintain information ontology.

```
<xsd:schema
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
        xmlns:jxb="http://java.sun.com/xml/ns/jaxb"
jxb:version="1.0">
...
<xsd:complexType name="KOBase">
   <xsd:sequence>
      <xsd:element name="Usage" type="xsd:string"/>
      <xsd:element name="Owner" type="xsd:string"/>
      <xsd:element name="KO" type="KnowledgeObject"
maxOccurs="unbounded"/>
   </xsd:sequence>
</xsd:complexType>

<xsd:complexType name="KnowledgeObject">
   <xsd:sequence>
      <xsd:element name="identifier" type="xsd:string"/>
      <xsd:element name="title" type="xsd:string"/>
      <xsd:element name="author" type="xsd:string"
           minOccurs="0" maxOccurs="unbounded"/>
... skip ...
   </xsd:sequence>
   <xsd:attribute name="shareable" type="xsd:boolean"
default="false"/>
</xsd:complexType>
</xsd:schema>
```

Figure 3. The part of XML schema of knowledge object base

Figure 4 shows the use cases diagram of parts of the generated Java classes in the UML. It is obvious that the generated Java classes are fully mapped to XML schema, and they provide complete and convenient methods to access KO base. The ObjectFactory class is a major Java class of JAXB, which is responsible for creating instances of schema-driven Java classes. The KOBase and KnowledgeObject classes are Java interfaces that specify the get/set methods to manipulate XML documents, and they are mapped to XML schema of knowledge object base. Because the generated Java classes can deal with XML documents, the system developer does not endeavor to write complex and tedious codes to handle KO base and can concentrate on domain processes.

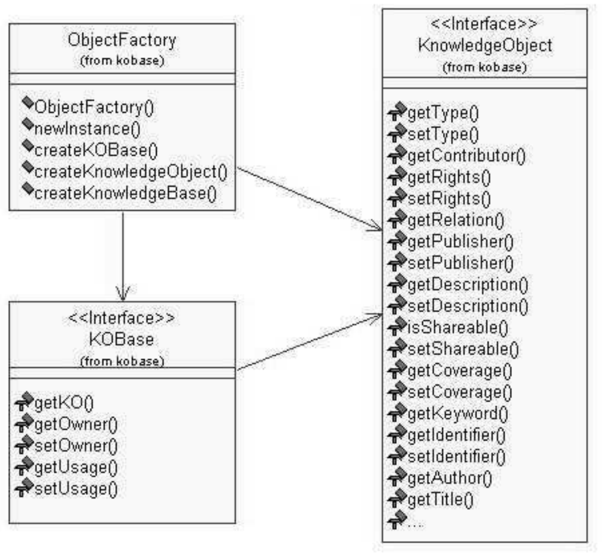


Figure 4. Use case diagram of information ontology maintainer

4.2 XML Binding Model

Since we can simply bind XML documents to in-memory objects, it is much easier to write XML-enabled programs. Figure 5 shows the framework of XML binding model in this study. The framework can be divided into three layers: domain application, Java objects, and XML documents. The mapping between Java objects and XML documents is relied on the JAXB technology. In JAXB technology, there are two major processes to take care of the translation between Java objects and XML documents: marshalling and unmarshalling processes. Marshalling Java objects means converting Java objects to XML documents. On the contrary, the process, turning XML documents to Java objects, is named unmarshalling process. After unmarshalling process, whole KOBase is mapped to in-memory Java objects, and the KOM components in domain application layer can manipulate XML documents through the in-memory KOBase and KnowledgeObject. Finally, marshalling process converts in-memory Java objects to XML format and stores in repository.

Although the binding model is efficient and convenient for manipulating XML documents, the KOM components lack flexibility to apply to other domains. The system developer must re-design the KOM components if we deploy the system to other industries. In order to reduce complication and error-proneness of development and to facilitate reusability of the system, we propose a scheme that can automatically generate the KOM components. The details will be discussed in next subsection.

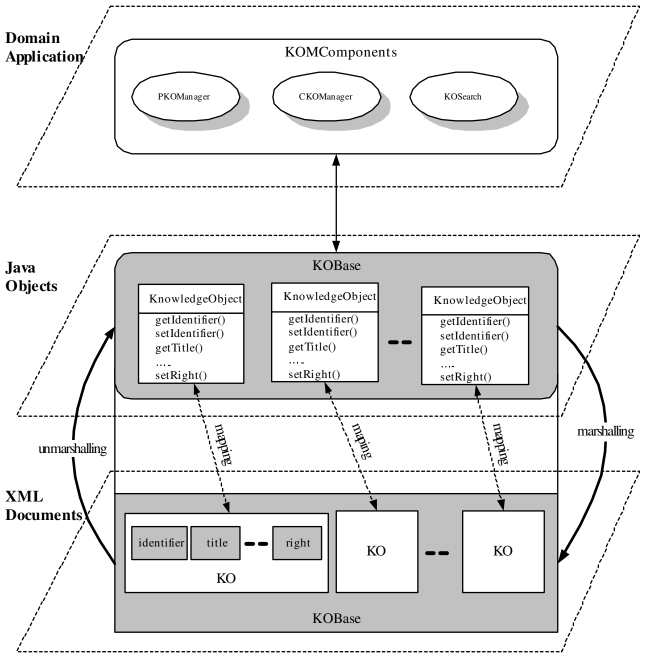


Figure 5. The Framework of XML Binding Model

4.3 A Scheme of Automatic Component Generation

The development of a new KMS is a time-consuming and laborious task, and the lack of generic KMS and the different characteristics of domain knowledge make it difficult to apply existing KMS to other industries. To remedy such limitations, we follow the design philosophy of component reusbility in developing the proposed KMS. In particular, we propose a scheme based on XML/Java technologies which can automatically generate the KO management (KOM) components so that one may reengineer the system to accommodate with different requirements of other domains.

Figure 6 shows the conceptual architecture of the proposed scheme. System developer defines a meta-schema description (MSD) for describing the format of the information ontology. The Component Generator reads the description of meta-schema via the MSD Parser, which is generated by JAXB, following, the parser binds to the description of meta-schema. The description of meta-schema that is a XML file contains the format of a KO, the description of JavaBean template, and the description of JSP template. The Component Generator, first, produces the Information Ontology XML Schema that defines format of information ontology, and then the Component Generator externally executes the JAXB compiler which will read the Information Ontology XML Schema for generating the Java source code of Information Ontology Maintainer. Next, the

Component Generator invokes the JBGenerator and the JSPGenerator according to the description. JBGenerator is responsible for reading JavaBean Template base and generating Java beans that are in charge of the management of knowledge object base. The knowledge object base, KOBase, is a repository whose contents are adherent to the XML format. The KOBase Manager manages the KOBase via the Information Ontology Maintainer component, and they are compiled into a Java package, named KOM Java Package which is to be deployed in the business logic layer mentioned in Figure 1. On the other hand, JSPGenerator is responsible for generating Java server pages that are the user interface in the presentation layer of Figure 1 according to the JSP Template base.

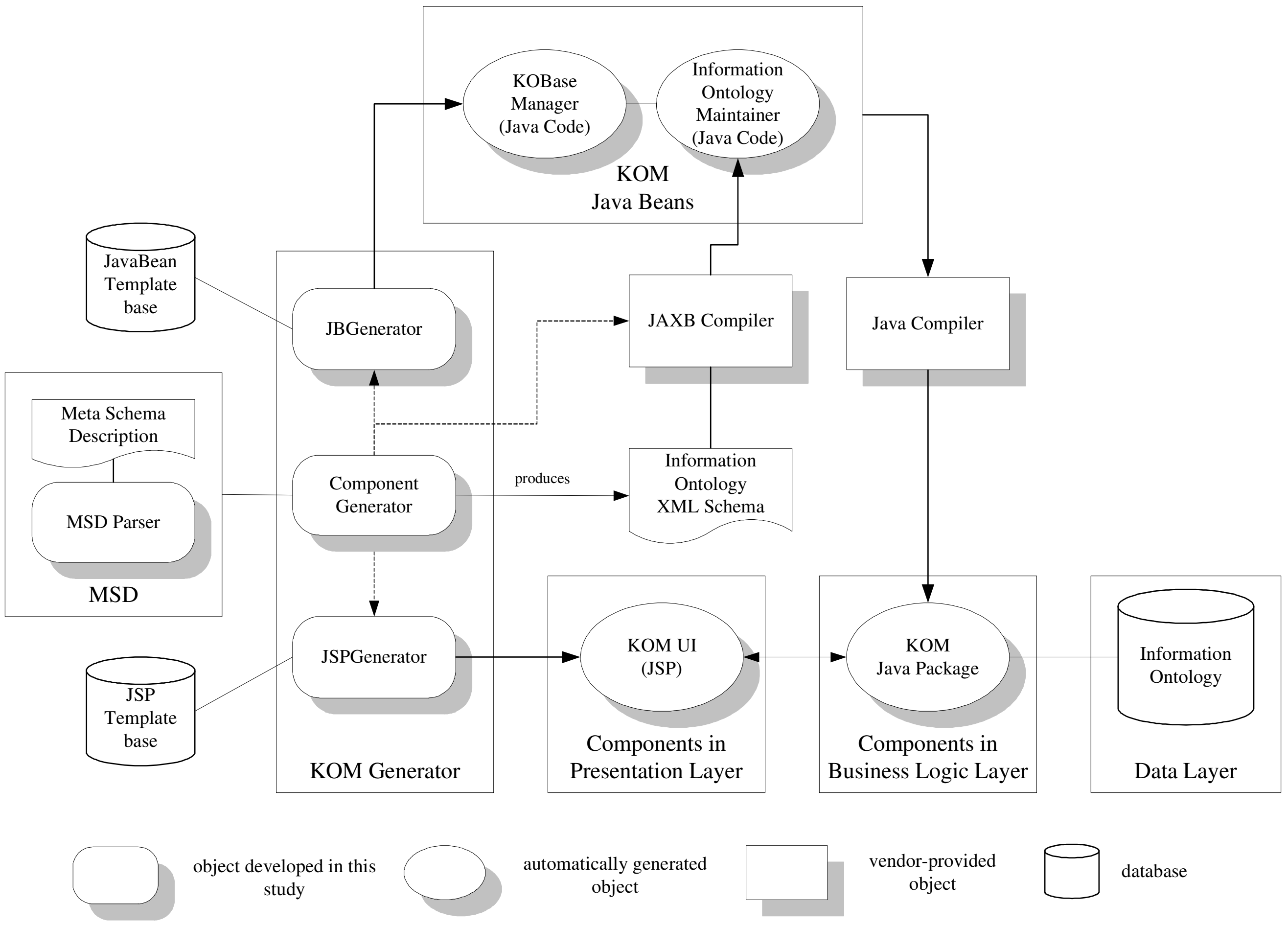


Figure 6. A scheme for automatic component generation

Figure 7 and 8 show an example of meta-schema description and JavaBean template. There are three major parts of meta schema description. First part is <InformationOntology> tag that specifies XML schema of information ontology. Second part describes what and where elements should be filled in. As shown in Figure 7, the Title, Author, and others elements should be filled in actions 1 and 3 in KOBean.tpl template file. An example of action is shown in Figure 8, which is a part of template code of KOBean.tpl. Last part is the description of JSP template that is similar to JavaBean template. Therefore, the system developer could only specify the meta schema description to automatically generate KOM components and then can reduce the complication and error-proneness of development.

```
<KOM>
  <InformationOntology>
        <Element name="identifier" type="string"/>
        <Element name="title" type="string"/>
        <Element name="author" type="string"
                    minOccurs="0"
                    maxOccurs="unbounded"/>
    .....
  </InformationOntology>
  <JavaBean>
    <Template filename="KOBean.tpl">
      <Action Id="1, 3">
        <Element>Title</Element>
        <Element>Author</Element>
        <Element> … </Element>
      </Action>
```

```
        </Action Id="2, 4">
           <Element> ... </Element>
        </Action>
        <ForAllElement>
           <Action Id="A">
              <Except> Identifier </Except>
           </Action>
           <Action Id="B"/>
        </ForAllElement>
     </Template>
  </JavaBean>
  <JSP>
     <Template filename="create.tpl">
     …
     </Template>
  </JSP>
</KOM>
```

Figure 7. The meta schema description of automatic component generation

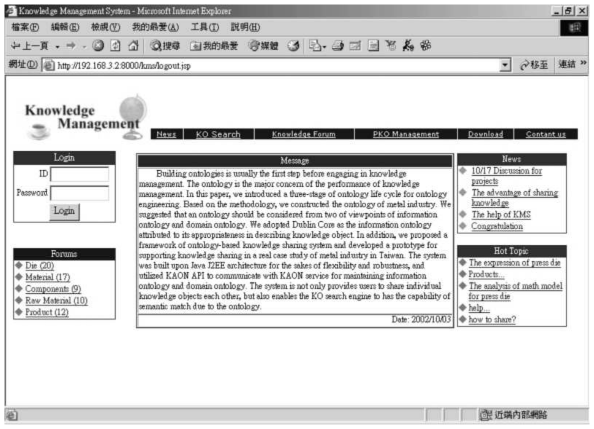
```
…
   public KOBean(KnowledgeObject ko) {
      koHashtable = new Hashtable();
      <Action id=1>
         putToHashtable("<Element>", ko.get<Element>());
      </Action>
      <Action id=2>
         putToHashtable("<Element>",
List2String(ko.get<Element>()));
      </Action>
      shareable = ko.isShareable();
   }
…
   <ForAllElements Id="B">
      public String get<Element>()  {
         return (String)koHashtable.get("<Element>");
      }
      public void set<Element>() {
         putToHashtable("<Element>", <Element>);
      }
      </ForAllElements>
…
```

Figure 8. A example of JavaBean template

5 System Demonstration

So far, we already developed a prototype of ontology-based knowledge management system, which runs on JBoss-3.0.3_Tomcat-4.1.12 EJB server in SUN Ultra 10 Workstation. The snapshots of system demonstration are shown in Figures 9, 10, and 11.

Figure 9 is the entry point of the system. User has to login the system first as an individual member or group member, and then start to engage in the activities of knowledge management including creating, editing, browsing, and searching knowledge objects. The creation of knowledge object with information ontology is shown in Figure 10, which is for system administrator. As Figure 10 indicates, user needs to select a knowledge object in advance, and then fills up the information ontology. Finally, the knowledge object and information ontology will be submitted to server and saved in common knowledge base and information ontology base.

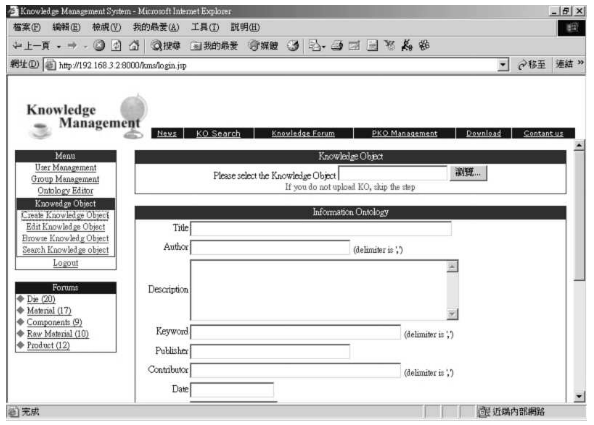


Message

Building ontologies is usually the first step before engaging in knowledge management. The ontology is the major concern of the performance of knowledge management. In this paper, we introduced a three-stage of ontology life cycle for ontology engineering. Based on the methodology, we constructed the ontology of metal industry. We suggested that an ontology should be considered from two of viewpoints of information ontology and domain ontology. We adopted Dublin Core as the information ontology attributed to its appropriateness in describing knowledge object. In addition, we proposed a framework of ontology-based knowledge sharing system and developed a prototype for supporting knowledge sharing in a real case study of metal industry in Taiwan. The system was built upon Java J2EE architecture for the sakes of flexibility and robustness, and utilized KAON API to communicate with KAON service for maintaining information ontology and domain ontology. The system is not only provides users to share individual knowledge objects each other, but also enables the KO search engine to has the capability of semantic match due to the ontology.

Date: 2002/10/03

Figure 9. The entry of knowledge management system



Knowledge Object

Please select the Knowledge Object

If you do not upload KO, skip the step

Information Ontology

Title

Author (delimiter is ',')

Description

Keyword (delimiter is ',')

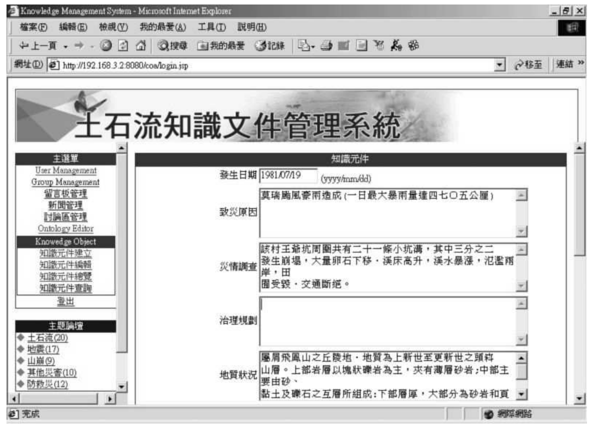
Publisher

Contributor (delimiter is ',')

Date

Figure10. Building a knowledge object with information ontology

As section 4 discussed, the system can be easy applied to other industry as shown in figure 11. It is obvious that information ontology is different between figure 10 and figure 11. Thus it can be seen that one may easy reengineer the system to accommodate with different requirements of other domains by proposed scheme in this study.



知識元件

發生日期 1981/07/19 (yyyy/mm/dd)

致災原因 莫瑞颱風豪雨造成(一日最大暴雨量達四七〇五公厘)

災情調查 該村王爺坑周圍共有二十一條小坑溝，其中三分之二發生崩塌，大量卵石下移，溪床高升，溪水暴漲，氾濫兩岸，田
園受毀，交通斷絕。

治理規劃

地質狀況 屬眉飛鳳山之丘陵地，地質為上新世至更新世之頭嵙山層。上部岩層以塊狀礫岩為主，夾有薄層砂岩;中部主要由砂、
黏土及礫石之互層所組成:下部層厚，大部分為砂岩和頁

Figure 11. Applying to other industry

6 Conclusions

In this study, we developed a KMS for metal industry in Taiwan. It is a difficult task to develop a KMS from scratch. In order to enhance the reusability of KMS, we proposed a scheme that can automatically generate KOM components to reduce the overhead in developing this KMS and increase its applicability to other domains. In addition, the XML binding model for knowledge object can facilitate the readability and exchangeability of KOs. That is to say, the KMS can easily get the information of KOs and does not arduously parse KOs, and it is very convenient to exchange KOs between KMSs.

7 Acknowledgements

This work presented was in part supported by Metal Industries Research & Development Center, Taiwan, under contract number 91B00021.

Reference

[1] J.-P. Barthés, "ISMICK and Knowledge Management," In J. F. Schreinemakers ed, Knowledge Management: Organization, Competence and Methodology, Proc. Of ISMICK'96, Rotterdam, the Neth., Wurzburg:Ergon Verlag, pp. 9-13, 21-22 October, 1996.

[2] A Abecker, A. Bernardi, K. Hinkelmann, O. Kühn, and M. Sintek, "Toward a Technology for Organizational Memories," IEEE Intelligent System, pp. 40-48, 13, May/June, 1998.

[3] L. Steels, "Corporate Knowledge Management," Proc. Of ISMICK'93, Compiégne, France, 1993, pp. 9-30

[4] A. Okunoye and H. Karsten, "IT as Enabler of Knowledge Management: Empirical Perspective from Research Organisations in Sub-Saharan Africa," 35th Annual Hawaii International Conference on System Sciences (HICSS'02), 04, 07-10 January, 2002.

[5] S. S. R. Abidi, "Knowledge Management in Healthcare: Towards 'Knowledge-Driven' Decision-Support Services," International Journal of Medical Informatics, 63, 2001, pp. 5-18

[6] J.-P. A. Barthès and C. A. Tacla, "Agent-Supported Portals and Knowledge Management in Complex R&D Projects," Computers in Industry, vol. 48 (1), 2002, pp. 3-16

[7] K.W. Chau, C. Chuntian, and C.W. Li, "Knowledge Management System on the Flow and Water Quality Modeling," Expert System with Applications, vol. 22 (4), 2002, pp. 321-330

[8] Y. Malhotra, "Why Knowledge Management Systems Fail? Enablers and Constraints of Knowledge Management in Human Enterprises," In Holsapple, C.W. (Ed.), Handbook on Knowledge Management 1: Knowledge Matters, Springer-Verlag, Heidelberg, Germany, 2002, pp. 577-599.

[9] H.R. Nemati, D.M. Steiger, L. S. Iyer, and R.T. Herschel, "Knowledge Warehouse: an Architectural Integration of Knowledge Management, Decision Support, Artificial Intelligence and Data Warehousing," Decision Support Systems, Vol. 33, 2002, pp. 143-16

[10] A. Waterson and A. Preece, "Verifying Ontological Commitment in Knowledge-based Systems," Knowledge-Based System, vol. 12, 1999, pp. 45-54

[11] A. Tiwana and B. Ramesh, "Integrating Knowledge on the Web," IEEE Internet Computing, Vol. 5, No. 3, 2001, pp. 32-39

[12] N. Guarino, "Understanding, Building, and Using Ontologies: A Commentary to Using Explicit Ontologies in KBS Development by van Heijst, Schreiber, and Wielinga," International Journal of Human and Computer Studies, 46, 1997, pp. 293-310

[13] Sun Microsystems Inc., Java Architecture for XML Binding. Available: http://java.sun.com/xml/jaxb

[14] The University of Karlsruhe, The Karlsruhe Ontology (KAON) tool suite. Available: http://kaon.semanticweb.org/

[15] Sun Microsystems Inc., The Java Architecture for XML Binding (JAXB) User's Guide. Available: http://java.sun.com/xml/docs.html