

Generating an XML-Based Search Index for an Effective Search of Office Documents

Somchai Chatvichienchai

*Associate Professor
Department of Information Media
Siebold University of Nagasaki, Japan
somchaic@sun.ac.jp*

Jinsan Lin

*Associate Professor
Faculty of Environment & Information Management
Nagoya Sangyo University, Japan
lin@agoya-sun.ac.jp*

Katsumi Tanaka

*Professor, Department of Social Information
Graduate School of Information
Kyoto University, Japan
ktanaka@i.kyoto-u.ac.jp*

ABSTRACT

Office applications are becoming a major pillar of today's organizations since they are used to edit a vast amount of digital documents. Finding these office documents in large databases that fit users' needs is becoming increasingly important. Broad one- or two-word searches in conventional search engines are often plagued by low precision, returning many irrelevant documents as their output. In order to solve this problem, we propose a technique that allows users to define search terms inside office documents and term descriptions presenting semantic relationships between the office documents and their search terms. This technique provides a means that generates an XML-based search index allowing users to effectively find out target office documents by search conditions, whose definitions are based on document types, search terms, and term descriptions. We also present the schema of the proposed search index that allows users to effectively search office documents of various types, and also present a search framework that uses the proposed technique.

Keywords: Digital documents, XML, template, schema, search index

1. BACKGROUND

Office applications, such as Microsoft Office Suites [16] and OpenOffice [17] etc., are becoming a major pillar of today's organizations since they are used to edit a vast amount of digital documents. As more and more office documents become electronically available, finding documents in large databases that fit users' needs is becoming increasingly important. Today, many organizations maintain a variety of office documents in a complex ad-hoc architecture that does not seem to fulfill the needs for companywide unstructured information in business processes, business functions, and the extended enterprise. To search documents and extract useful information out of them, we need efficient and effective means of organizing and indexing the documents. In the past, the document search problem was dealt with using the database query approach [9][11][12] or the text-based search approach [1].

In this paper, a *search term* inside an office document denotes a text string or a phrase that users employ as a reference to identify the target office documents. Since almost all conventional search systems do not provide a means that defines the relationships between office documents and their search terms, these search systems return a list of files containing the search terms as their outputs. The search results, therefore, mostly contain irrelevant documents. Consider a sample of a meeting minute shown in Figure 1.

Meeting Minute		Secret Level:	High
Type of Meeting:	Director Meeting		
Date:	2006-11-15	Time:	10:00-10:45
Attendees:	Anny Dowson	Managing Director	
	John Brown	Accounting Director	
	Bill Brinson	Personnel Director	
Agenda Title:	Company Annual Profit Report		
John reported that the profit of company of this year will be 20% higher than the target. Anny wants to give special bonus to the outstanding employees.			
Finally Anny requested John to present the bonus plan in the next director meeting that will be held on next week.			
Submitted by	Bill Brinson	Approved by	John Brown

Figure 1. Sample of a Meeting Minute

Suppose that Bill wants to find out minutes of the meetings that Anny Dowson had attended and discussed on the topic of company annual profit. As the search result, almost all conventional search programs basically output a list of the document files that contain the terms “Anny Dowson” and “Company Annual Profit.” These search programs do not provide a means to search the meeting minutes that contain the term “Anny Dowson,” which is regarded as a meeting attendee, and the term “Company Annual Profit,” which is regarded as the agenda title. One solution for this problem is to write metadata (such as document type, meeting date, attendees, agenda title, file path locating physical addresses of the documents, etc.) into database when meeting minutes are created and updated. This solution has a drawback, however, in that users have to manually generate metadata by copying search terms from office documents and defining the file addresses of the office documents. This approach leads to errors, inaccuracies, and duplications of data reentry [24].

The objective of this paper is to propose a technique that collects the search terms from office documents, and generates a search index that can effectively identify the target office documents by the search conditions whose definitions are based on document types, search terms, and term descriptions. In this paper, we focus on the digital documents that are edited by Microsoft Office Word 2003 (*MS Word*, for short), since it is widely used in enterprise and government organizations. However, the proposed technique can be applied to the documents edited by Microsoft Office Word 2007 with some trivial technique modification.

In this paper, we use the term *office documents* to denote the digital documents edited by MS Word. As XML [21] has become a de facto standard also for describing web and office documents [13], we use XML for defining a search index of office documents. Therefore, the search index can be easily imported to existing search engines. The search index is designed to effectively address various document types (such as meeting minutes, sale reports, contracts, letters, etc.). Users can easily build a search query by defining the search terms and add operators such as equals, greater than, or less than in the list or between to further expand or restrict the search scope. The contribution of this paper can be summarized as follows.

1. We introduce a technique that allows users to define search terms inside the office documents created from a *document template* (*template*, for short) by creating an XML schema that defines the search terms of the template and making a mapping from these search terms to elements of the XML schema.
2. We present schema of the search index that is designed to effectively locate office documents of various types.
3. We propose an office document search framework that employs the proposed technique.

The rest of the paper is organized as follows. Related work is discussed in section 2. Section 3 presents basic concepts of XML, XML schema, and XPath. In section 4, we discuss the issues on generating a search index for office documents. In section 5, we present the proposed office document search framework in detail. Section 6 presents our conclusions.

2. RELATED WORK

Recently, there has been a surge of interest in topics of searching local files driven by Google Desktop Search [8] and others [3][25] who now offer desktop search engines. These systems work by creating an index of the files found on a computer (or network) and allowing the user to perform keyword searches across the data. These systems, while powerful, do not give the users any input on how their files will be organized and presented; so, even though users can find files that contain information on "Profit Report," for example, they do not have a way of describing how the word "Profit Report" is regarded as a data value of the field "Agenda Title" in these files. Our work, however, provides a way to describe it.

The work by Jenkins C. and Inman D. [10] proposes a technique for automatically generating qualified Dublin Core metadata [6] on a web server using a Java Servlet [18]. The metadata is structured using the Resource Description Framework (RDF) [23] and expressed in XML. The description covers 10 out of 15 standard metadata. The metadata elements are title, creator, subject, description, publisher, date, format, identifier, relation, and rights. When the URL of a document is passed to the servlet, it harvests the title, date, and format from the HTML tags. The metadata description is represented by an abstract, which is the first 25 words found in the body of the document. The metadata subject is represented by a series of keywords. The keywords are extracted by parsing the actual content. The metadata relation is used to represent a resource that is hyper-linked from the current resource. However, this work focuses on Web documents which are described by HTML. Then, this work cannot be applied with office documents.

The Lifestreams System [7] also presents the users with a non-directory-based collection of the documents. The system presents the users with a collection of documents based on the time the document was added to the system or received by the user. Basically, the system chooses one property of the file to be the key on which all navigation occurs. Since the types of documents are defined by system file properties, this system cannot fulfill the complicated search conditions of today's organizations.

The relational model has been proposed as a means to provide universal access to data [5]. Systems for "schema-agnostic" keyword searches on databases, such as DBXplorer [2], BANKS [4] and Discover [9], model a response as a tree connecting nodes (tuples) that contain the different keywords in a query (or more generally, nodes that satisfy specified conditions). Here "schema-agnostic" means that the queries need not use any schema information (although the

evaluation system can exploit schema information). However, these work focus on relational databases while our work focuses on office documents which are regarded as unstructured data.

3. BASIC CONCEPTS

This discussion of basic concepts covers XML (3.1), XML schema (3.2), and XPath (3.3).

3.1. XML

XML provides a way to describe structured data. Unlike TML tags, which are primarily used to control the display and appearance of data, XML tags are used to define the structure and data types of the data itself. XML uses a set of tags to delineate elements of data. An element contains sub-elements and attributes. These sub-elements in turn contain other sub-elements, and so on. Elements that contain sub-elements or carry attributes are said to have complex types, whereas elements that contain numbers (and strings, and dates, etc.) but do not contain any sub-elements are said to have simple types. Some elements have attributes; attributes always have simple types. As XML is simple, platform-independent, and a widely adopted standard, XML data is adopted throughout an organization and across the organizations. Figure 2(a) depicts a sample XML document in a file called *minute.xml*, which represents a part of the meeting minute from Figure 1. This XML document consists of a main element: *minute*, and the sub-elements: *secretLevel*, *meetingType*, *date*, *time*, *attendees*, *agendaTitle*, and *submittedBy*. An XML document can be viewed as a tree-structure (a node tree), with the elements and attributes defined as nodes. A sample tree structure of a document from Figure 2(a) is shown in Figure 2(b).

3.2. XML Schema

XML schemas [22] are documents that are used to define and validate the content and structure of XML documents, just as a database schema defines and validates the tables, columns, and data types that make up a database. An XML schema defines and describes certain types of XML data by using the XML schema definition language (XSD). XML schema elements (elements, attributes, types, and groups) are used to define the valid structure, valid data content, and relationships of certain types of XML data. XML schemas can also provide default values for attributes, and elements.

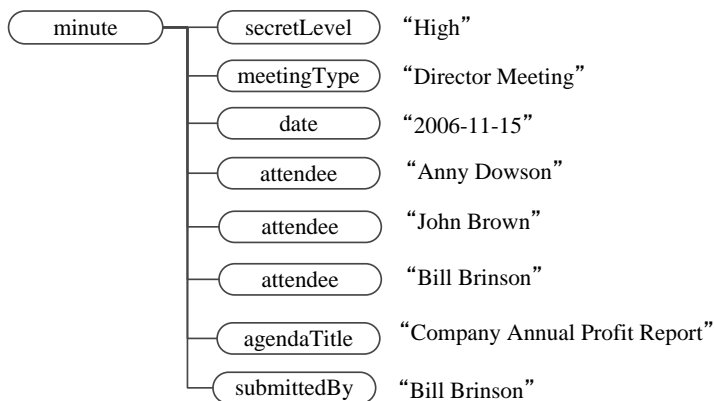
Figure 3(a) depicts a sample XML schema in a file called *Minute.xsd* for XML data of Figure 2. Each element in the schema has a prefix *xsd:* which is associated with the XML schema namespace [19] through the declaration: *xmlns:xsd="http://www.w3.org/2001/XMLSchema,"* which appears in the schema element. The prefix *xsd:* is used by convention to denote the XML schema namespace, although any prefix can be used. The same prefix, and hence the same association, also appears on the names of built-in simple types; e.g. *xsd:string*.

```

<?xml version="1.0" encoding="UTF-8" standalone="no" ?>
<minute>
  <secretLevel> High </secretLevel>
  <meetingType> Director Meeting </meetingType>
  <date> 2006-11-15 </date>
  <attendee> Anny Dowson </attendee>
  <attendee> John Brown </attendee>
  <attendee> Bill Brinson </attendee>
  <agendaTitle> Company Annual Profit Report </agendaTitle>
  <submittedBy> Bill Brinson </submittedBy>
</minute>

```

(a) An XML document in a file called *Minute.xml* representing a part of meeting minutes of Figure 1



(b) A tree structure for XML document of Figure 2(a)

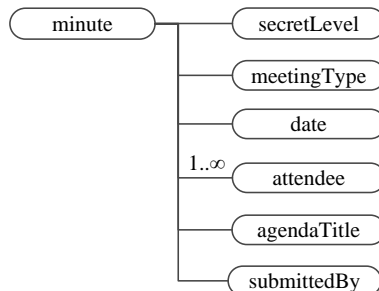
Figure 2: Sample XML Document and Its Tree Structure

The purpose of the association is to identify the elements and simple types as belonging to the vocabulary of the XML schema language rather than the vocabulary of the schema author. For the sake of clarity in the text, we mention just the names of elements and simple types (e.g., *simpleType*), and omit the prefix. The definitions of complex types in the meeting minute schema declare sequences of elements that must appear in the instance document. The occurrence of individual elements declared in the so-called content models of these types may be optional, as indicated by a 0 value for the attribute *minOccurs*, or be otherwise constrained depending on the values of *minOccurs* and *maxOccurs*. The XML schema also provides constraints that apply to groups of elements appearing in a content model. Note that the constraints do not apply to attributes. A tree structure of the sample schema is presented in Figure3(b). Note that the

occurrence of an element is described as a label of the edge between the node of that element and its parent node. The label “1..∞” denotes that the occurrence is more than zero. The label “0..∞” denotes that the occurrence is zero or more than zero. The edge with no label denotes that the occurrence of element node under its parent node is one.

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  elementFormDefault="qualified">
  <xsd:element name="secretLevel" type="xs:string"/>
  <xsd:element name="meetingType" type="xs:string"/>
  <xsd:element name="date" type="xs:date"/>
  <xsd:element name="attendee" type="xs:string"/>
  <xsd:element name="agendaTitle" type="xs:string"/>
  <xsd:element name="submittedBy" type="xs:string"/>
  <xsd:element name="minute">
    <xsd:complexType mixed="true">
      <xsd:sequence>
        <xsd:element ref="secretLevel"/>
        <xsd:element ref="meetingType"/>
        <xsd:element ref="date"/>
        <xsd:element ref="attendee" minOccurs="1"
          maxOccurs="unbounded"/>
        <xsd:element ref="agendaTitle"/>
        <xsd:element ref="submittedBy"/>
      </xsd:sequence>
    </xsd:complexType>
  </xsd:element>
</xsd:schema>
```

(a) An XML schema in a file called *Minute.xsd*



(b) A tree structure for the schema in *Minute.xsd*

Figure 3. Sample of an XML Schema and Its Tree Structure

3.3. XPath

XPath [20] is an expression language for addressing portions of an XML document or for computing values (strings, numbers, or Boolean values), based on the content of an XML document. Location paths of elements and attributes can be *absolute* or *relative*. An absolute location path, prefixed by a '/' character, starts from the root of the XML tree. A relative location path, which starts with an element name, describes a path whose initial point is any element in the document. Furthermore, we can define conditions of locating an element and attribute by predicates of XPath. For example, path expression */minute/agendaTitle/text()* locates text string "Company Annual Profit Report" of element *agendaTitle*.

4. ISSUES ON GENERATING A SEARCH INDEX

Our goal is to generate a search index that effectively identifies the office documents that are satisfied by the search conditions whose definitions are based on document types and their search terms. Here, three main issues on generating the search index are discussed:

- How to identify search terms inside office documents
- How to collect search terms after users save the document files
- Properties of the search index that can apply to various documents

4.1. How To Identify Search Terms Inside Office Documents

We observe that office documents are often referred to as unstructured and that their internal formats strongly vary on versions of MS Word. We suggest that methods of identifying search terms should not vary on internal formats of office documents so that these methods remain effective when the software version changes. In order to satisfy this requirement, we propose a method of identifying the search terms of office documents by using the XML map function of MS Word. This method uses the MS Word's feature that allows users to associate one or more XML schemas with an office document and then map all or some of the schema elements to search terms of the office document. XML schemas in Word are called *XML structures*. In order to unify the format of office documents of the same type, we propose to use a template, which is a document pattern used to create a new document. Instead of having users re-create a document over and over from scratch, a template allows users to keep their page formatting and "fill in the blanks" that need changing. Each template contains document settings such as fonts, page layout, special formatting, and styles.

For each document type, *system manager* (who is the administrator of this office-document-searching system) creates a template and designs an XML schema that defines search terms of that document type. The system manager attaches the XML schema to the template, and uses the XML structure task pane of MS Word to map the search terms of the template to elements of the attached schema. Based on our experiment of creating new documents from the template,

we notice that these new documents inherit this mapping definition from the template. Therefore, the benefit of mapping search terms of the template to elements of the attached schema is to cut off the time and labor cost in defining this mapping for each new document. We describe this method in detail in the next section.

4.2. How To Collect Search Terms after Users Save the Document Files

Based on Visual Basic Application macro [14], we develop a macro that automatically outputs the mapped search terms of an office document into an XML file. Since this macro is a short and simple program, the system manager can easily customize it to match the running environment (e.g., document type and directory locations of the outputted XML files, etc). The system manager embeds the customized macro in the template and makes it activate when the document file is saved. Based on our experiment of creating new documents from the template, we notice all macros of the template are automatically copied to the new documents. When the office document created from the template is saved, the macro will automatically output the search terms into *search term instance file* of XML type. The search index generation program is executed by the system manager to read the *search term instance files* of the same document type and creates/updates sub-index for that document type.

4.3. Properties of Search Index Applicable to Various Types of Documents

In order to reduce the cost of developing the office document search system, we designed a search index that can locate office documents of various types. The schema of this search index (Figure 4) holds the following properties.

- The search index consists of sub-indices, each of which contains information for locating the documents of the same type.
- Each sub-index contains template information, search term schema, semantic descriptions of search terms, and information (such as search terms and file addresses) for locating office documents.
- Search terms are classified into non-iterative and iterative types. In addition, search terms of the iterative type are categorized into groups. A search term is stored in element “term” whose attribute “DescID” denotes the corresponding description element holding a semantic description of that search term. For example, as shown in Figure 5, the search term “attendee” is classified into iterative type and grouped under “attendees.”

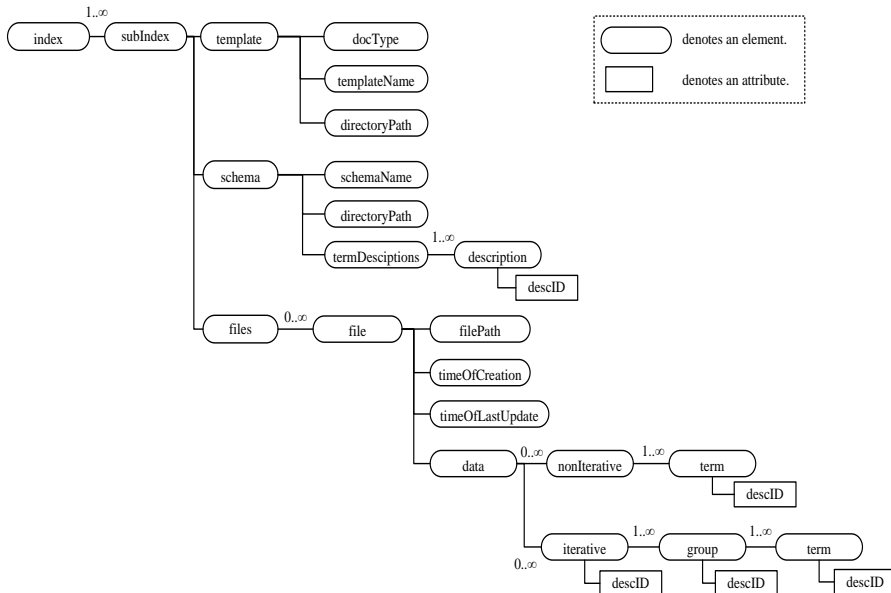


Figure 4: Tree Structure of Schema of the Proposed Search Index

5. AN OFFICE DOCUMENT SEARCH FRAMEWORK

Figure 6 depicts the system architecture of the proposed framework. This system consists of six processes:

- Template design
- Search term schema definition
- Description definition of search terms
- Document creation/modification
- Search index generation
- Document search

5.1. Template Design: For each document type, the system manager creates a template that is used in the organization to create a new document of that type. Figure 7 depicts a sample template *MeetingMinute.dot* for creating meeting minutes. This template is used as a blank meeting minute and users get to fill in all the boxes. It has orders, boxes, lines, and instructions to tell users where they may enter important texts. The templates designed at this subsystem are passed to the next process before releasing to the users.

```

<?xml version="1.0" encoding="UTF-8"?>
<index xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="Search_Index.xsd">
  <subIndex>
    <template>
      <docType>Meeting Minute</docType>
      <templateName>MeetingMinute.dot</templateName>
      <directoryPath> http://www.sun.ac.jp/offices/templates</directoryPath>
    </template>
    <schema>
      <schemaName>MeetingMinute.xsd</schemaName>
      <directoryPath>http://www.sun.ac.jp/offices/schemas</directoryPath>
      <termDescriptions>
        <description descID="secretLevel">Secret Level</description>
        <description descID="meetingType">Type of Meeting</description>
        <description descID="date">Meeting Date</description>
        <description descID="attendees">Attendees</description>
        <description descID="attendee">Attendee</description>
        <description descID="agendaTitle">Agenda Title</description>
        <description descID="submittedBy">Submitted by</description>
      </termDescriptions>
    </schema>
    <files>
      <file>
        <filePath> http://www.sun.ac.jp/meeting/Meeting_Minute.doc</filePath>
        <timeOfCreation>2007/01/15 10:15:12</timeOfCreation>
        <timeofLastUpdate>2007/01/17 15:20:45</timeofLastUpdate>
        <data>
          <nonIterative>
            <term descID="secretLevel">High</term>
            <term descID="meetingType">Director Meeting</term>
            <term descID="date">2006-11-15</term>
            <term descID="agendaTitle">Company Annual Profit Report</term>
            <term descID="submittedBy">Bill Brinson</term>
          </nonIterative>
          <iterative>
            <group descID="attendees">
              <term descID="attendee">Anny Dowson</term>
              <term descID="attendee">John Brown</term>
              <term descID="attendee">Bill Brinson</term>
            </group>
          </iterative>
        </data>
      </file>
    </files>
  </subIndex>
</index>

```

Figure 5. Sample Search Index File

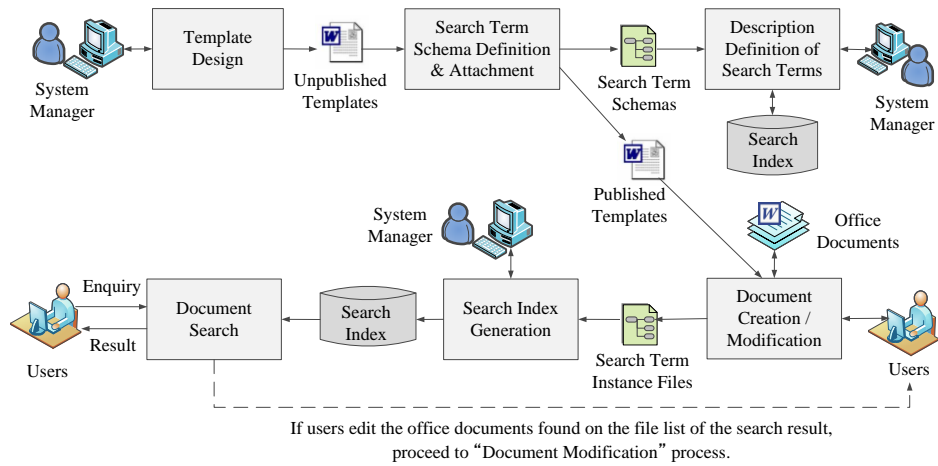


Figure 6. Architecture of the Office Document Searching System

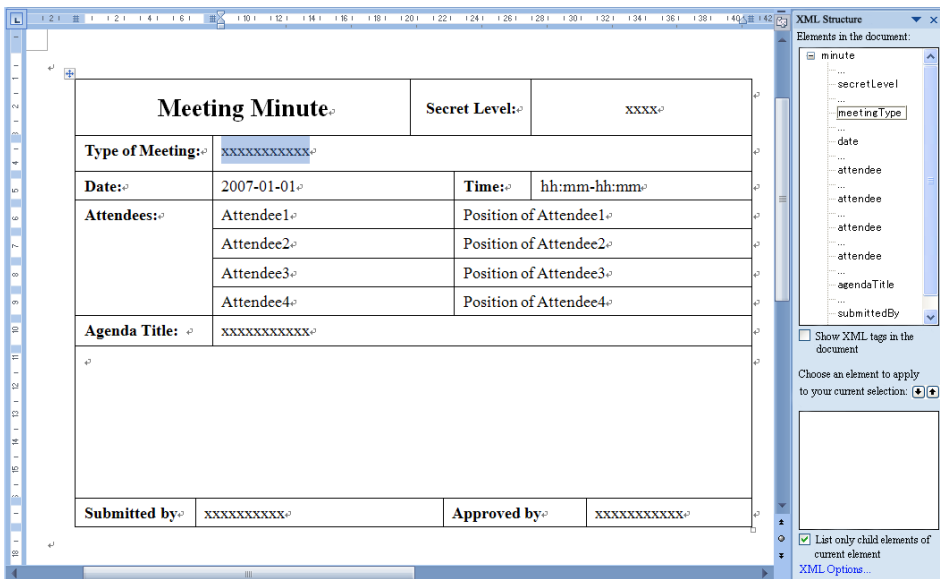


Figure 7. Sample of Document Template to Which XML Scheme Is Attached

5.2. Search Term Schema Definition / Attachment: The system manager defines a *search term schema* that describes search terms of the office documents created from the template of the previous process. In other word, the system manager defines a search term schema for each document type. A sample search term schema for meeting minute documents is shown earlier in Figure 3(a). This schema has *secretLevel*, *meetingType*, *date*, *attendee*, *agendaTitle* and *submittedBy* used for specifying conditions of searching the target meeting minutes. The next step is that the system manager defines a mapping from search terms of the template to elements of the corresponding search term schema. MS Word has a *Schema Library* to which the system manager can add multiple search term schemas. Once a search term schema within the Schema Library is attached to a template, the system manager can use a visual interface to graphically map schema elements from the *XML structure task pane* (see the right frame of Figure 7) to the template itself. The templates with attached search term schemas are released to users for creating new documents. The system manager embeds the customized macro into the template and makes it activate when the new document is saved. The macro automatically outputs the mapped search terms into an XML file (called the search term instance file).

5.3. Description Definition of Search Terms: Since lengths of element names of search term schemas are rather short, it may not be suitable to use these for describing search terms of office documents. In order to solve this problem, we propose a process that allows the system manager to define *term descriptions* that present the semantic relationships between office documents and their search terms. Term descriptions are displayed at the search query form to guide users to enter the corresponding search terms. In order to make users familiar with term descriptions, we recommend that the terms be defined with those that appear in office documents. A sample screen for defining descriptions of search terms of meeting minutes is shown in Figure 8. The term description definition program requests the system manager to input the file address of the search term schema. Based on the specified search term schema, the program displays element names of the search term schema and allows the system manager to input document type, file address of the template, and term descriptions. The information inputted by the system manager is saved as a part of search sub-index of the specified document type. Figure 5 illustrates a sample search index that includes sub-index of meeting minutes.

5.4. Document Creation / Modification: In this system, users create new office documents from the templates of the previous process. The text strings or phrases that are defined as search terms of office documents are automatically outputted into a search term instance file of XML type when the office documents are saved.

Description Definition of Search Terms

Template Schema File Path	w:\offices\schemas\MeetingMinute.xsd ...
Template File Path	w:\offices\templates\MeetingMinute.dot ...
Document Type	Meeting Minute

Non-iterative Type

No.	Schema Element Name	Term Description
1.	secretLevel	Secret Level
2.	meetingType	Type of Meeting
3.	date	Meeting Date
4.	agendaTitle	Agenda Title
5.	submittedBy	Submitted by

Iterative Type

Group Name		Attendees
No.	Schema Element Name	Term Description
1.	attendee	Attendee

Figure 8. Sample Screen of Term Definitions for Search Sub-Index

5.5. Search Index Generation: The search index generation program gathers search term instance files to create and update the search index. As the search index is described in XML format, it can be easily imported to other search engines. Note that scope of index generation can be defined to a file or directory unit(s). Furthermore, the program execution schedule can be defined by the system manager. The index generation program has two index generation modes: *full generation* and *differential generation*. The full generation mode is the starting point for index generation, and produces a search index of all the office documents that are selected. A differential generation mode updates the index of all files that have changed since the last generation. The advantage of a differential generation is that it shortens index generation time compared with a full generation.

5.6. Document Search: In order to attain high precision search results, a search condition is defined by document type and its search terms. The document search program requests users to specify the document types that the users want to search. Based on the document type specified by users, the document search program displays the search request form for that document type. In the search request form of a document type, users can easily build a document search by defining search terms, and add operators such as equals, greater than, less than, in the list or between to further expand or restrict the search scope. The search program will transform the query defined at search query form into query defined by XPath. The XPath queries return file paths of the office documents satisfied by the search conditions. Two search query samples that are based on the sample search index shown in Figure 5 are depicted as follows.

Example 1: Search all meeting minutes that are submitted by Bill Brinson. This query can be expressed by the following XPath query.

```
//subIndex[./template/docType/text()='Meeting
Minute']//file[./data/nonIterative/term
[@descID='submittedBy' and contains(./text(),'Bill Brinson')]]/filePath
```

Example 2: Search all director meetings that Anny Dowson attended. This query can be expressed by the following XPath query.

```
//subIndex[./template[./docType='Meeting
Minute']]/file[./data/nonIterative/term
[@descID='meetingType' and ./text()='Director Meeting']
and ./data/iterative/term
[@descID='attendee' and contains(./text(),'Anny Dowson')]]/filePath
```

6. CONCLUSIONS AND FUTURE WORK

As more and more office documents become electronically available, finding these documents in large databases that fit users' needs is becoming increasingly important. In order to satisfy this requirement, we have proposed a technique that generates an XML-based search index for the documents created from templates. A template is a prescribed form file that is used when creating a new document. This technique requires that a system manager generate an XML schema defining search terms of a document type, and make semantic mapping from these search terms of the template to elements of the XML schema, and create a macro program into the template. In order to copy search term mapping information and the macro program from the template to new documents, this technique requires that users create new documents from the designed templates. When users save the documents created from the designed templates, the macros inside the documents automatically output search terms of the documents to

XML files. The search index generation program collects these XML files and generates search index of XML format.

We have presented schema of the search index that is designed to effectively locate office documents of various types. The search index consists of sub-indices, each of which is used to locate documents of the same type. Each sub-index contains necessary information (such as template's file path, search term schema, search terms, semantic descriptions of the search terms, and file addresses of office documents, etc.) for finding out the target office documents. Based on its properties, the search index enables users to effectively find out the target office documents by the search conditions, whose definitions are based on document types, search terms, and term descriptions. Since the search index is described in XML format, it can be easily imported to other search engines. We have also proposed an office document search framework that uses the proposed technique. The search framework consists of document template design, search term schema definition / attachment, document creation / modification, search index generation, and document search processes.

This work leaves three future work issues. The first is to extend the proposed technique to make it possible to search the spreadsheets edited by spreadsheet programs. Since users can create tables in multiple worksheets of a spreadsheet, we need to consider how to design schema of search terms of these tables and how users specify search terms of the target tables in the document query. The second is to develop a prototype of the search index generation system. Response time of search query, which varies on search conditions and search index size, is our main investigation target. The third is to construct a centralized index that can be shared with users of organizations without disclosing any confidential or private information. In order to apply access control at query runtime, we need to consider how to integrate information on user access right into the centralized search index.

ACKNOWLEDGMENTS

The authors thank the anonymous referees for their valuable comments and suggestions to improve the presentation of the paper.

REFERENCES

- [1] Agichtein, E., Cucerzan, S. 2005. Predicting accuracy of extracting information from unstructured text collections. *In* Proc. of 4th International Conference on Information and Knowledge Management (CIKM 2005), Bremen, Germany, 413-420.
- [2] Agrawal, S., Chaudhuri, S., Das, G. 2002. DBXplorer: A system for keyword-based search over relational databases. *In* Proc. of 18th International Conference on Data Engineering (ICDE2002), San Jose, California, USA, 5-16.

- [3] Apple. (2007). Apple Computer, Tiger Preview: Spotlight, Available at <http://www.apple.com/macosx/features/spotlight/>.
- [4] Bhalotia, G.; Hulgeri, A.; Nakhe, C.; Chakrabarti, S.; and Sudarshan, S. 2002. Keyword searching and browsing in databases using BANKS. *In Proc. of 18th International Conference on Data Engineering (ICDE2002)*, San Jose, California, USA, 431-440.
- [5] Codd, E. 1970. A relational model of data for large shared data banks, *Commun. ACM* 13(6).
- [6] DCMI (Dublin Core Metadata Initiative). 2006. DCMI Metadata Terms, <http://dublincore.org/documents/dcmi-terms/>.
- [7] Freeman, E., and Gelernter, D. 1996. Lifestreams: A storage model for personal data. *SIGMOD Record (ACM Special Interest Group on Management of Data)*, 25(1), 80-87.
- [8] Google. 2006. Google Desktop Search, <http://desktop.google.com/>.
- [9] Hristidis, V., and Papakonstantinou, Y. 2002. DISCOVER: Keyword search in relational databases. *In Proc. of 28th International Conference on Very Large Data Bases (VLDB2002)*, Hong Kong, China, 670-681.
- [10] Jenkins, C., and Inman, D. 2000. Server-Side Automatic Metadata Generation using Qualified Dublin Core and RDF. *Kyoto International Conference on Digital Libraries 2000*: 245-253.
- [11] Masermann, U., and Vossen, G. 2000. Schema independent database querying (on and off the Web). *In Proc. of International Database Engineering and Applications Symposium (IDEAS2000)*, Yokohoma, Japan, 55-64.
- [12] Masermann, U., and Vossen, G. 2000) Design and implementation of a novel approach to keyword searching in relational databases. *In Proc. of East-European Conference on Advances in Databases and Information Systems Held Jointly with International Conference on Database Systems for Advanced Applications Symposium (ADBIS-DASFAA)*, Prague, Czech Republic, 171-184.
- [13] Microsoft. 2003. Microsoft Office 2003 XML Reference Schemas. Available at: <http://www.microsoft.com/office/xml/default.mspx>.
- [14] Microsoft. 2003) Microsoft Word 2003 Visual Basic Applications Reference. [http://msdn2.microsoft.com/en-us/library/aa272078\(office.11\).aspx](http://msdn2.microsoft.com/en-us/library/aa272078(office.11).aspx).
- [15] Microsoft. 2005. Visual Studio 2005 Tools for Office. Available at: [http://msdn2.microsoft.com/en-us/library/aa167897\(office.11\).aspx](http://msdn2.microsoft.com/en-us/library/aa167897(office.11).aspx).
- [16] Microsoft. 2006. Microsoft Office Suites. Available at: <http://office.microsoft.com/en-us/suites/FX101677751033.aspx>.
- [17] OpenOffice. 2007. OpenOffice Suite Version 2.1. Available at: <http://www.openoffice.org/index.html>.
- [18] SDN (SUN Developer Network). 2006. Java Servlet Technology. Available at: <http://java.sun.com/products/servlet/>.
- [19] W3C. 1999. XML Namespaces Recommendation. Available at: <http://www.w3.org/TR/1999/REC-xml-names-19990114/>.
- [20] W3C. 1999. XML Path Language (XPath) Version 1.0. Available at: <http://www.w3.org/TR/xpath>.
- [21] W3C. 2000. Extensible Markup Language (XML) 1.0 (Second Edition). Available at: <http://www.w3c.org/TR/REC-xml>.

- [22] W3C. 2001. XML Schema. Available at: <http://www.w3c.org/XML/Schema>.
- [23] W3C. 2004. Resource Description Framework (RDF). Available at: <http://www.w3.org/RDF/>.
- [24] Wei, C.P.; Hu, P.J.; and Dong Y.X. 2002. Managing document categories in e-commerce environments: an evolution-based approach, *European Journal of Information Systems*, (11) 208-222.
- [25] Yahoo. 2006. Yahoo Desktop Search. Available at: <http://desktop.yahoo.com/>.

ABOUT THE AUTHORS:

Dr. Somchai Chatvichienchai is an associate professor at Siebold University of Nagasaki, Japan. His research interests include database theory and systems, XML, Access Control, and Web information retrieval. He is a member of the ACM, IEEE, the Database Society of Japan (DBSJ) and the Information Processing Society of Japan (IPSJ).

Dr. Jinsan Lin is an associate professor at Nagoya Sangyo University, Japan. His research interests include engineering of organizational intelligence, customer relationship management, and ubiquitous network society. He is a member of the Operations Research Society of Japan, the Japanese Society of Management Information, and the Japan Society for Social Informatics.

Dr. Katsumi Tanaka is a professor at Kyoto University, Japan. His research interests include database theory and systems, Web information retrieval, and multi-media content retrieval. He is a member of the ACM, IEEE, the Database Society of Japan (DBSJ), and the Information Processing Society of Japan (IPSJ). He is currently a vice president of DBSJ and the fellow of IPSJ.