Supporting visual access to a distributed organizational memory warehouse in the web environment

Yunan Song
Iowa State University

Follow this and additional works at: http://lib.dr.iastate.edu/rtd
Part of the Computer Sciences Commons

Recommended Citation

This Thesis is brought to you for free and open access by Digital Repository @ Iowa State University. It has been accepted for inclusion in Retrospective Theses and Dissertations by an authorized administrator of Digital Repository @ Iowa State University. For more information, please contact hinefuku@iastate.edu.
Supporting visual access to a distributed organizational memory warehouse in the web environment

by

Yunan Song

A thesis submitted to the graduate faculty in partial fulfillment of the requirements for the degree of MASTER OF SCIENCE

Major: Computer Science

Program of Study Committee:
Leslie Miller, Major Professor
Sree Nilakanta
Shashi K. Gadia

Iowa State University
Ames, Iowa
2006

Copyright © Yunan Song, 2006. All rights reserved.
# TABLE OF CONTENTS

LIST OF FIGURES ........................................................................................................... iii
LIST OF LISTINGS ........................................................................................................ iv
LIST OF TABLES ........................................................................................................... v

ABSTRACT ......................................................................................................................... iv

CHAPTER 1. INTRODUCTION ........................................................................................... 1

CHAPTER 2. RELATED WORK ......................................................................................... 3
  2.1 Basic concepts ....................................................................................................... 3
  2.2 Main challenges .................................................................................................... 5
  2.3 Topic maps ........................................................................................................... 6
    2.3.1 Topic maps basic concepts ............................................................................ 6
    2.3.2 Topic maps visualization and navigation ....................................................... 7
    2.3.3 Advantages and disadvantages .................................................................... 8
  2.4 Knowledge management tools ............................................................................. 8
  2.5 Embedding existing tool issues ........................................................................... 9

CHAPTER 3. MODEL ....................................................................................................... 10
  3.1 Overview ............................................................................................................... 10
  3.2 Object views ....................................................................................................... 12
  3.3 Topic maps .......................................................................................................... 12
  3.4 Data types .......................................................................................................... 13
  3.5 Data sources ....................................................................................................... 14
  3.6 Internal object data store .................................................................................... 14
  3.7 Request processor ............................................................................................... 14

CHAPTER 4. PROTOTYPE AND IMPLEMENTATION ....................................................... 15
  4.1 Prototype ............................................................................................................. 15
    4.1.1 Topic map web interface ............................................................................. 15
    4.1.2 Email tools .................................................................................................. 20
  4.2 Implementation ................................................................................................... 24
    4.2.1 Ajax and Google Web Toolkit ...................................................................... 24
    4.2.2 Apache Struts ............................................................................................. 27
    4.2.3 Main classes ............................................................................................... 29

CHAPTER 5. CONCLUSION AND FUTURE WORK ....................................................... 31

CHAPTER 6. REFERENCES ............................................................................................. 32
LIST OF FIGURES

Figure 3.1 Block diagram of the distributed organizational memory data warehouse model 11
Figure 3.2 Source node layout and request/data flow for retrieval 14
Figure 4.1 Screen of initial page and home page 16
Figure 4.2 Screen of the Search by Memory Collection 18
Figure 4.3 Screen of Email 18
Figure 4.4 Screen of initial page when magnifying icon is clicked on 19
Figure 4.5 Screen of general search result 21
Figure 4.6 Screen of after launching Frequency Analysis Tool 22
Figure 4.7 Screen of Frequency Analysis Result 23
Figure 4.8 Screen of popup emails 23
LIST OF LISTINGS

Listing 4.1 code for Ajax implementation 26
Listing 4.2. part of struts-config.xml 29
Listing 4.3. Code for frequencyAnalysis class 30
Table 1 Main classes

29
Organizational memories play a significant role in knowledge management but several challenges confront their use. Artifacts of organizational memory are many and varied. Access and use of the stored artifact is influenced by the user’s understanding of these information objects as well as their context. In our work, we use a topic map to represent user cognition of contextualized information. Topic maps allow for access and analysis of stored memory artifacts. We implement the topic map with both a Java client interface and a web interface to access the organizational memory warehouse. The thesis also presents the design and development of this organizational memory warehouse with several simple tools with web access via topic maps. The running warehouse example uses email as the data type.
1 INTRODUCTION

Today an organization has to be able to make use of all of its available information in order to be competitive. In particular researchers have noted that organizational knowledge may be accumulated and retained using several organizational memory systems [1,2,3]. Advances in information technologies have helped in accumulating knowledge but the paradigms applicable to collecting and storing the knowledge do little to motivate its use by managers and decision makers. Accessing and using organizational memories, have been a challenge because of the multifaceted nature of memories and knowledge sources. Moreover, characteristics of the decision tasks where the knowledge and memory support are needed, pose interesting challenges in designing and developing knowledge management systems.

Currently, research in organizational memory management mostly deal with the creation, integration, maintenance, dissemination, and use of all kinds of knowledge within an organization [4,5]. Recent challenges to developing successful organizational memory systems fall into three primary areas: managing the knowledge, generating and using the knowledge, developing practical systems. We will address these challenges in detail in the related work chapter. Our work is focuses on the third challenge, and we present a model of a distributed organizational memory data warehouse. During our research, we found out that a topic map can be used to enhance the visual use of a distributed organizational memory warehouse. We describe the use of topic maps to navigate the network of organizational memory and employ appropriate knowledge management and analysis tools. Email is used as the data type in our test for evaluating the proposed organizational warehouse system and our implementation in web environment. We also claim that our topic map web interface is extendable to add other existing knowledge management and analysis tools so that our goal in the future is to make the current system a framework. One limitation of our current implementation is that it only works in Mozilla browser after version 1.5.

In the next chapter we look at the related work and issues of developing and using
organizational memory. In Chapter 3 we describe the model of our distributed organizational memory data warehouse. Chapter 4 looks at the prototype and implementation of this system based on our model. Chapter 5 concludes the thesis and suggests the possibilities for future work.
2 RELATED WORK

This chapter discusses related work. Section 2.1 gives an introduction of some basic concepts of knowledge management, organizational memory and organizational memory systems. Section 2.2 presents the main challenges existing in current research on these issues. Section 2.3 looks to the concept of topic maps along with their advantages and disadvantages. Section 2.4 looks at the knowledge management tools we implemented to test our prototype. Section 2.5 looks at the issue of utilizing existing analysis tools in our prototype.

2.1 Basic concepts

Knowledge management (KM) is an essential capability in the emerging knowledge economy [6]. KM refers to a range of practices used by organizations to identify, create, represent, and distribute knowledge for reuse, awareness and learning across the organization. The key distinction made by the majority of knowledge management practitioners is the distinction between implicit and explicit knowledge [6]. The implicit knowledge (also known as tacit knowledge) is often subconscious, internalized, and the individual may or may not be aware of what he/she knows and how he/she accomplishes particular results. At the opposite is explicit knowledge - knowledge that the individual holds explicitly and consciously in mental focus, and may communicate to others. Another way of denoting the distinction is that implicit knowledge is in our heads, and explicit knowledge is what we have codified.

Nonaka and Takeuchi [7] argued that a successful knowledge management program needs to, on one hand, convert implicit knowledge into explicit codified knowledge in order to share it, but also individuals and groups need to internalize and make codified knowledge personally meaningful. A knowledge organization is one in which the key asset is knowledge, including both the implicit and explicit assets.

Organizational memory (OM) is a stored, interrelated collection of organizational history reflected among the many parts [8]. It includes both stored records
and tacit knowledge and covers the various facets of organizational tasks, employees, and their task environments [9, 10, 11, 12]. Because it can grow rapidly and become a vast repository of information and knowledge, several researchers have recognized the import of this organizational memory in effecting organizational performance [13, 14, 15, 16, 17]. Ackerman and Halverson [18], however, take a critical view of prior research on OM and argue for a theoretical base to properly define and empirically validate future research. They state that as socio-technical systems, organizations and their memories conform to social structures and norms while employing technical models. They use the theory of distributed cognition to develop a theoretical foundation for organizational memory. The basic tenets of this theory are that knowledge evolves from a community of practice and that cognition and inferences result from the shared meaning among the participants (hence the distribution) [19]. Communities of practice fulfill a number of functions with respect to the creation, accumulation, and diffusion of knowledge in an organization through exchange and interpretation of information, by retaining knowledge, by stewarding competencies, and providing homes for identities [20]. Collective thinking creates knowledge that otherwise would not be evident. Using empirical data and qualitative methods, Ackerman and Halverson [18] illustrate application of the theory of distributed cognition to validate the use of organizational memory in decision making.

This thesis presents the design and development of an organizational memory warehouse that supports the distributed cognition theory. We have adopted the design-science research paradigm to present our system design. Hevner et. al [21] argue that design-science research must addresses important unsolved problems in unique or innovative ways or solve problems in more effective or efficient ways. Design-science research proponents identify six nominal process steps, namely problem identification and motivation, objectives of a solution, design and development, demonstration, evaluation, and communication [22]. The first four process steps are considered suitable for doing design research. In addition, when systems are built for testing and validating theory, the appropriate entry point for design research is indeed the design and
development process. At this stage, an artifact is built and the underlying theory is tested using the artifact. The main purpose is to yield a proof of concept of the theory and the system that supports it.

An **Organizational memory system** (OMS) is built from traditional information technologies, such as databases and telecommunications links. The intent is to use this “superstructure” provided by information technology to capture and disseminate knowledge within the organization. Mandiwalla et al. [24] define an organizational memory system to include a database management system that can represent more than transactional data and an application that runs on top of the database management system. They further describe the generic requirements of an OMS to include different types of memory, including how to represent, capture, and use organizational memory. Nemati et al. [1] illustrate that a knowledge warehouse combines three abilities: an ability to efficiently generate, store, retrieve and, in general, manage explicit knowledge in various forms; an ability to store, execute, and manage the analysis tasks and their and their supporting technologies with minimal interaction and cognitive requirements from the decision maker; an ability to update the knowledge warehouse via a feedback loop of validated analysis output. In the next section the main challenges in building current organizational memory systems are discussed.

### 2.2 Main challenges

As was introduced in first chapter, current challenges facing organizational memory systems fall into three categories [25]. We will discuss these challenges here.

The first type of challenge is managing the knowledge. This type of challenge primarily exists in managing informal and formal knowledge. This is one of the most obvious challenges facing the design and implementation of an organizational memory system. As we know, informal knowledge is wild and hard to capture. In order to solve this difficulty, Conklin [23] suggests that organizations shift their view of knowledge from its current artifact-oriented perspective towards a more process-oriented view.
The second type of challenge is related to knowledge workers. Knowledge workers play an important role in developing, implementing, using and maintaining an organizational memory. The main difficulties are when and what knowledge needs to be captured, and how to capture this knowledge, and how to save and maintain the knowledge.

The last type of the challenge is the gap between organizational memory system research and application development. So the best way to develop the system for a given organization is to avoid changing the operations of that organization. Our organizational memory warehouse system focus on this technical goal after we get our organizational goal settled.

2.3 Topic maps

As social and linguistic dimensions of knowledge become pertinent, users need appropriate tools to support the search and analysis of knowledge. Topic maps are one such tool.

Topic Maps are defined by the ISO standard 13250 [26] which allows the description of knowledge and links to existing information resources. They provide a bridge between the domains of knowledge representation and information management by building a semantic network above information resources, which allows users to navigate at a higher level of abstraction. However, Topic Maps are multidimensional knowledge and can hence build a very large semantic network. Therefore, to visualize and navigate a topic map is essential and interesting. We first look at the basic concept of topic maps and then we discuss the visualization and navigation issues.

2.3.1 Topic maps basic concepts

Topic Maps have been described as the “GPS of the information universe” [27], as they are designed to enhance navigation in complex data sets.
Topic Maps consists of topics and tools that are connected by associations and which point to information resources through occurrences. A topic is a syntactic construct which corresponds to the expression of a real-world concept in a computer system, that is, it can be anything. For example, in Figure 4.1, each oval is a topic. Topic maps provide a subject based classification of resources where each resource represents a real world object or a tool. In topic maps, three constructs are provided for describing the subjects represented by the topics: names, occurrences, and associations. These describe the names, properties, and relationships of subjects, respectively. A name may be assigned to more than one topic and a topic may have more than one name. In addition by defining scope and types a name can become rich and complex. An occurrence links to one or more real knowledge sources. An occurrence, however, is not part of the topic map. Associations describe relationships among topics and are independent of the real knowledge objects. Associations add value through the relationships. An object in topic map would be a knowledge object such as a memory collection or a resource (e.g., an email analysis tool) that acts on the information object (e.g., an email collection).

2.3.2 Topic maps visualization and navigation

Topic Maps are powerful but they may be complex. Therefore, the user interfaces may significantly reduce the cognitive load of users when working with these complex structures. So visualization is an essential technique in large information spaces to provide navigation facilities. According to Gershon et al. [28], it also enables people to use a natural tool of observation and processing – their eyes as well as their brain – extract knowledge more efficiently and find insights. Therefore, the goal of Topic Map visualization is to help the user to locate relevant information quickly and explore the structure easily. Our implementation of the topic map interface is based on the work of Schneiderman [29], “the visual information-seeking mantra is: overview first, then zoom and filter, then details on-demand”.

2.3.3 Advantages and disadvantages

Smolnik and Erdman [30] state that topic maps provide strong paradigms and concepts for the semantic structuring of link networks and therefore, they are a considerable solution for organizing and navigating large and, continuously growing network of organizational memories. Topic maps provide a subject based classification of resources where each resource represents a real world object. In topic maps, three constructs describing the subjects represented by the topics: names, occurrences, and associations can describe the names, properties, and relationships of subjects, respectively. A name may be assigned to more than one topic and a topic may have more than one name. In addition by defining scope and types a name can become rich and complex. An occurrence links to one or more real knowledge sources. An occurrence, however, is not part of the topic map. Associations describe relationships among topics and are independent of the real knowledge objects. Associations add value through the relationships. An object in topic map could be a knowledge object such as a memory collection or a resource that acts on the information object. For example, an email analysis tool is a knowledge object that operates on the email collections. However, the limitation is the topic map does not provide a separation of the declarative part of the topic map from its instance, thus making it difficult to design and maintain the schema and the instance.

2.4 Knowledge management tools

Topic map is not only used to navigate within the domain of topics, but also navigate within the domain of topics with operations. Here operations can be tools based on topics. So in our topic map structure, tools are also seen as topics, but special topics that can trigger actions on the workflow of topics. In this section, we will look at the tools we developed for our organizational memory warehouse systems and discuss how it works in this scenario.
In our initial feasibility tests, we used email as the data type. One of the tools developed for emails is a frequency analysis tool based on a vector space search. The vector space model [31] sees all documents, queries and profiles as being represented by term vectors. The full system consists of three parts: indexing the documents, weight assignment and retrieval or filtering. During the retrieval phase each document and committee profile is represented by a list of weighted terms and stored in an inverted file. Retrieval is based on similarity and return matched list. So when doing an analysis, we use this vector space mechanism to get the list of relevant emails from our data sources.

2.5 Embedding existing tool issues

Our idea was to provide a framework that can be extended by a user to embed existing tools instead of developing everything from scratch. We have looked into some techniques and tools to integrate existing tool into our organizational memory system. We focused on the problem of how to make a standalone program run in the web environment. When we want to integrate an existing program, this program can either be available in source code or an executable code. The dilemma is that when we have the source code, we haven’t found an efficient way to separate the view part and controller part of the program. This is a critical issue since in a web environment the view part must run on the client side and interact with the action or controller part on the server. As a result, we need to separate the view part and manually integrate the controller part into our system. Only having the executable part of a program we want to integrate faces the same problem. Therefore, we expect that the web browser interface will be more limited than the java client approach.
3 MODEL

The goal of any organization is to make optimal use of the business knowledge that it has accumulated over the years. To do this, it is necessary to determine the sources of business knowledge and provide members of the organization access to it in a timely matter. Here we focus on the latter issue of providing access and assume that business knowledge sources are employee knowledge and artifacts (both internal and external).

The proposed environment has been designed to provide support for accessing and analyzing the collective memory of the organization. We start by providing a simplistic overview of the model and then briefly examining the important components in more detail.

3.1 Overview

An organization’s memory can be seen as consisting of a space that includes knowledge of the expertise of current employees, artifacts, and the tools required to locate and/or interpret available information. The memory space is distributed across the organization and beyond.

Any organizational memory warehouse must be able to deal with the distributed nature of the memory space and be able to fully integrate the memory’s information with the necessary tools. To deal with this problem, we introduce a distributed organizational data warehouse.

We see our contribution as being the development of an environment that provides a complete and easy to use solution to making the most of an organization’s memory. The distributed format of the proposed warehouse provides a realistic mapping into the organizational memory space that is typical in organizations. Perhaps more important, the proposed model is highly scalable.

A block diagram of the complete model is shown in Figure 3.1. Artifacts and information about the expertise of current employees can either be stored in the data
sources or in the Internal Object-Oriented Data Store and Toolkit. Artifacts in the data sources can be stored in any format to take advantage of existing software or compression techniques. To simplify communication between components and provide for standardization within the model, object views are used to define data types. Data from the data sources are converted into object view instances using local interface views [33].

A topic map that supports visual access to the organization’s memory space via a combination of search terms and tool interfaces serves as the user’s entry point to the warehouse. Use of the topic map interface provides the user with an easy to use mechanism for defining search terms and choosing the appropriate interpretation/analysis/learning tools.

The remainder of this section briefly examines the main components of the model.

![Block diagram of the distributed organizational memory data warehouse model.](image)
3.2 Object views

The object view type [33] is defined as being an extension of the object model (EOM). The views have a traditional object structure (attributes and methods) with the restriction that they support a derivation method. The derivation method is used to generate the public and private attributes of each object instance created through a local interface view.

The individual data sources are expected to have local control. The local interface view is a view [33] object type that is used by the local data administrator to provide a mechanism to make the local data accessible to the proposed model in object form. The local interface allows distribution transparency and representation transparency, while hiding or converting (mapping) some of the data from the data source. The local interface view belongs to the data source. It interacts directly with the data source and passes the result to the wrapper which controls communication with other components of the proposed model. A given data source and its wrapper can support multiple local interface views in order to present its data in different ways to different applications or users.

3.3 Topic maps

A topic map is used in the model to provide users with visual access to both search and analysis. It is this interesting blend of semantic search and analysis that motivates its use in the model.

Here we define a topic map to be the directed acyclic graph \( T = (N,E) \), where \( N \) is the set of topic map terms, data types, or search/analysis tools. The directed edges in \( E \) are of the type \((n1,n2)\), where \( n1 \) is a topic map term or a data type and \( n2 \) is a topic map term, a data type, or a search/analysis tool. An implicit node, called the root, points to the topic map terms, data types and/or tools that make up the first level of the topic map.
Nodes with out degree zero are said to be leaves of the topic map. Leaves point to the information in the organization’s memory and carry any search terms and/or data types accumulated on the path form the root of the topic map to the leaf. Note that leaves that are either search terms or data types make use of an implicit search tool. In addition each non-leaf level of the topic map has a search tool that can be used to initiate a search based on the topic map terms that have been traversed to get to the topic level.

Topic maps are presented to the user via either the Web Interface or the Java Client Interface depending on the access mode chosen by the user. Regardless of the user’s access mode the visual structure and use of the topic map interface remains unchanged. The primary difference between the modes is in the details of their respective implementations.

3.4 Data types

The data types supported will depend to some extent on the nature of the organization. Typically, the data types will include the people that the organization relies on for expertise and the artifacts that make up the organizations’ memory. In the implementation described in Section 4, the data types used are the typical artifacts, e.g., emails, meeting minutes, reports, and presentations. Extending the supported data types is simply a matter of ensuring that the appropriate tools for search and analysis are available in the set of supported tools.

3.5 Data sources

A block diagram of the transformation of a data source is shown in Figure 3.2. The query is translated to the appropriate request string and passed to the local interface view (LIV). The LIV passes the request string to the data source and converts the response into the objects defined by the LIV view type. The set of result view objects are
then passed back to the Data Source Access Module and cached in the Internal Object Data Store for analysis.

![Diagram](image)

Figure 3.2 source node layout and request/data flow for retrieval.

### 3.6 Internal object data store

The Internal Object Data Store is based on our earlier work on object-oriented data warehouses [32]. The Internal Object Data Store combines tools and the cached data to allow in depth analysis of artifacts. The tool set consists of a set of built in (kernel) tools for managing tools and data and a set of user added tools for creating and analyzing data [34].

### 3.7 Request processor

The request processor receives a Request object from the User Interface Controller that defines the search criteria and the result format required to complete the request. The tool type and action required is built into the class structure by employing the Polymorphic design pattern. Each tool that is added either implicitly or explicitly to the topic map requires the definition of a class that extends the Request class. These new classes add any additional information/actions required by the tool.
4 PROTOTYPE AND IMPLEMENTATION

This chapter illustrates the prototype and implementation of the topic map web user interface and the tools. Section 4.1 will look at the prototype of the topic map web user interface and some tools we have developed. We will present the functionality of the prototype by walking through some simple examples of its use. The implementation details and main techniques will be introduced in section 4.2.

4.1 Prototype

To illustrate the feasibility of the proposed model, a prototype has been developed. In this chapter, we look at some real examples to illustrate the feasibility and functionality of our topic map web interface and email tools. Section 4.1.1 looks at the prototype while section 4.1.2 presents the email tools using in this prototype.

4.1.1 Topic map web interface

The topic map web interface has been implemented using Ajax [36] to simplify its use and add dynamic features. As a result, the topic map portion of the page remains unchanged when tools are initiated and added to the current web page. We will illustrate this feature in section 4.1.2. The initial screen of the topic map web interface is composed of two parts: a fixed toolbar on the top of the screen and a main panel of the topic map in the middle of the screen. By saying “fixed”, it means the toolbar is never changed and is in control of the main panel. Figure 4.1 shows the initial screen encountered by a user via his/her web browser.
The topic map is shown in the main panel on the page. As we introduced in section 4 of chapter 2, topic map is visualized with a DAG structure. The topic map is composed of topic nodes which are shown as either ovals or rectangles. We use oval and rectangle to differentiate the type of topic nodes. Ovals are topic nodes that represent topics while rectangles represent tools. Ovals can have child nodes which can either be ovals or rectangles but rectangles can only be leaf nodes. We call one view of the topic map one level. Clicking on a node sends the user to a deeper level in the path. The initial screen in Figure 4.1 is the top level of the topic map, and all nodes in this screen are topic nodes which may have ovals or rectangles as child nodes in the next level.

Taking a closer look at the toolbar on the top of the screen, we find five icons which have the normal meanings. Home, back, forward -- these first three icons from the left side in the toolbar are used for navigation of the topic map. The magnifying glass icon is used for a keyword search within the domain according to the path label for user’s current topic map search. The path label contains the history of the topic nodes user has
clicked to get to the current level. The question mark icon is used for help.

Navigation in the topic map is flexible and can be controlled either by clicking on a topic map node (oval or rectangle) or using the navigation icons in the toolbar. Clicking on an oval moves the user to the next level in the topic map. Clicking on the home icon in the toolbar takes the user back to the first level of the topic map as shown in figure 4.1. Clicking on the back icon in the toolbar allows the user to move back to the most recently visited topic map level back along the path label. The exception is when user moves to current top level page by clicking home icon, then the previous level according to the path label is the deepest level of the topic map in previous visiting period. When the user has moved back to the beginning point which is always the initial topic map screen, or the user hasn’t moved after launching the topic map, the topic map system will give the user an alert message. Clicking on a forward icon in the toolbar moves the topic map to a deeper level. The actual forward move depends on the user’s previous operation. If the previous operation was the result of clicking either the home or the back icon, clicking on the forward icon will return the user to his/her previous location in the topic map. If the previous operation was a click on a topic map node, then clicking on the forward icon will give the user an alert which means that the user hasn’t been to the next level of current topic map. Clicking on home icon or a topic map node will add home or the topic node the user clicked on to the path label. And clicking back or forward icon is using the path label to judge which topic map level to locate.

Clicking on Search Memory Collection node in Figure 4.1 will move to next level of topic map as shown in Figure 4.2. The path label now is: Home, Search Memory Collection and the pivot of the path label is pointing to Search Memory Collection. Clicking on the back icon in Figure 4.2 will move the topic map to the upper level which is the initial topic map page in Figure 4.1. The path label now is not changed except that the current pivot is pointing to home now. Clicking forward icon now will move to the next level which is Figure 4.2 and the current pivot of path label is pointing to Search Memory Collection again. If we click on the Email icon shown in Figure 4.2, it will
move to next level of topic map as shown in Figure 4.3 and the path label now is: Home, Search Memory Collection, and Email. The pivot now is pointing to Email.

Figure 4.2 Screen of the Search by Memory Collection

Figure 4.3 Screen of Email
Now let’s take a look at the magnifying glass icon in toolbar of the topic map. This icon represents a general keyword search clicking on the icon generates a new screen that allows the user to use a vector space search of the organizational memory artifacts. The search tool screen uses the current topic map path label to set up current conditions. Figure 4.4 is the initial page when the magnifying glass icon is clicked. The general search tool page is composed of two parts. We will first go through the left part which is a set of input forms.

![Figure 4.4 Screen of initial page when magnifying icon is clicked on](image)

In the input form part, the first input box on the top is used to input a list of keywords separated by commas. The second input part is a drop down list which is used to select data type to set as criteria during the search. This selection is pre-set according to the current path label when clicking on the magnifying glass icon. But the user can still change to another data type by resetting the pull down menu. Basically the query request that was sent to the server contains keywords list, data type, and time interval. So
the rest part of the input form on the left side is two ways to select time interval. If the user doesn’t select “Time Interval Based on Current Date” nor “Select Specific Time Interval”, the query will search the whole memory collection of the selected type. If the user decides to search with a time interval, he/she can either choose select an interval based on current date or select specific time intervals with starting date and ending date. The right side of the search page is used to display results dynamically. Initially, there are no results so there is a zero below the Contract All and Expand All buttons. Figure 4.5 shows an example when doing general search. Suppose we click on magnifying glass icon after clicking on topic node Email in figure 4.2 and the path label will have the content: Home, Search Memory Collection, and Email. The Select Data Type is automatically set to Email based on the current path label. Figure 4.5 shows the result when the list of keywords is: interest, and the select time interval has been set to be last 10 days. After clicking on the Search button, the result returned is an email list with only the From and Subject information. The emails can be expanded to get the whole content. In Figure 4.5, as with our testing emails, the result consists of two emails. The minus sign in front of the email title means this email has already been expanded showing the body of the email. The plus sign in front of the second email title means this email is contracted and only shows the title information. The user can click the minus and plus sign to switch between an expanding and contracting mode. Also we can use the button Expand All and Contract All on the top to toggle between the two different display modes. After a search, clicking Clear button will clean all input fields to get ready for a new round of searching.

The next subsection looks at a specific email tool developed for our tests.

4.1.2 Email tools

When we walk through the topic map web interface, the tree structure topic map nodes are either ovals or rectangles. We have seen in the previous chapter that an oval
represents the topics and rectangles represent tools. This chapter will discuss the main features of rectangles which are tools in our topic map web interface. Figure 4.3 shows the topic map level that is created by clicking on the Email node. The level consists of one oval and five rectangles which all are tools related to analyzing emails.

![Figure 4.5 Screen of general search result](image)

Clicking on the Email Frequency Analysis tool rectangle in Figure 4.3 opens up the input screen shown on the lower left corner of the web page, see Figure 4.6. The tool allows the user to choose a set of keywords used to search the email collection, the start and stop dates of the time interval that the user is interested in and the number of intervals that the user wants the total interval divided into. Clicking on the View Charts button generates the bar graph shown in the lower right hand corner of the screen as shown in Figure 4.7.
The result shown in Figure 4.7 is a label showing the time interval along with a bar showing the number of emails in this interval. The button within each bar with a number is clickable. Clicking on it will populate a new screen with the email list from this time interval. For example, in Figure 4.7, the result has eleven emails in total and in the last interval, there are two emails showing in the yellow bar. When clicking on the button within this bar, it will populate the two emails in this time interval, see Figure 4.8. Same as the keyword search tool, each email can be switched between the expanded and contracted mode.
Topic Map Organizational Memory

Business Domain

Cluster By Content

Communication Graph

Social Network

Analyze

Frequency Analysis

Figure 4.7 Screen of Frequency Analysis Result

Figure 4.8 Screen of popup emails
4.2 Implementation

In this section, we introduce the main techniques used during implementation along with their advances and limitations. Subsection 4.2.1 looks at Ajax and the Google Web Toolkit which we used to add a dynamic flavor to the web interface. Subsection 4.2.2 introduces Apache Struts which is a framework we used to implement Model-View-Controller design pattern. Subsection 4.2.3 looks at the main data structures during implementation.

4.2.1 Ajax and Google Web Toolkit

The web browser topic map interface has been implemented using Ajax to simplify its use and add a dynamic flavor. As a result, the input page and the resulting bar graph for the Email Frequency Analysis tool are added to the same page and the status of the topic map portion of the page remains unchanged. The Google Web Toolkit [37] has been used to generate the JavaScript that supports the generation of the topic map interface and the navigation within the topic map. Both the General Keyword Search Tool (triggered by clicking on the magnifying glass icon) and Email Frequency Analysis Tool implement this technology. Let’s take the Email Frequency Analysis Tool as the example. When the user clicks on the Frequency Analysis rectangle in Figure 4.3, the only change on the web page is that a new frame of the frequency analysis tool input form is added to the lower left corner of the web page. The user can still navigate or call a General Keyword Search Tool from the topic map because the topic map’s functionality is still active. When the view charts button is clicked, the result bar graph is added to the right bottom dynamically according to the data returned from the server. The color of each bar is determined at run time by the range of the number of emails in each time interval. What’s more, before a user clicks on any of the colored button in bar chart to populate emails, the email data is already sent to the browser waiting to be
displayed. In this way, the response time is improved. The underlying technology is Ajax.

Ajax is the term shorthand for Asynchronous JavaScript and XML. The basic technologies involved in Ajax applications are:

- HTML is used to build Web forms and identify fields for use in the rest of the application.
- JavaScript code is the core code running Ajax applications and it helps facilitate communication with server applications.
- DHTML, or Dynamic HTML helps you update your forms dynamically.
- DOM, the Document Object Model, will be used through JavaScript code to work with both the structure of the HTML and XML returned from the server.

The use of Ajax is based on the use of the XMLHttpRequest component which is a JavaScript object. It puts JavaScript technology and the XMLHttpRequest object between the web form and the server. When the user fills out a form, the data is sent to JavaScript code rather directly to the server. Instead, the JavaScript code grabs the data from the form and sends a request to the server. And the request is sent asynchronously which means the JavaScript code doesn’t wait for the server to respond, instead, the JavaScript code defines what to do with the data sent back.

In our implementation, the keyword search tool and the frequency analysis tool were implemented using Ajax. For example, when the user submit a keyword list, time intervals and the number of interval in frequency analysis tool, the JavaScript code behind it instantiates the XMLHttpRequest object appropriately depending on the browser, and then opens a request to the supplied URL. When the state of the request changes, the processStateChange() function will be called (which displays the bar chart with the data returned). See Listing 4.1.
function retrieveURL(url) {
    if (window.XMLHttpRequest) { // Non-IE browsers
        req = new XMLHttpRequestQ ;
        req.onreadystatechange = processStateChange;
        try {
            req.open("GET", url, true);
        } catch (e) {
            alert(e);
        }
        req.send(null);
    } else if (window.ActiveXObject) { // IE
        req = new ActiveXObject("Microsoft.XMLHTTP");
        if (req) {
            req.onreadystatechange = processStateChange;
            req.open("GET", url, true);
            req.send();
        }
    }
}

function processStateChange() {
    if (req.readyState == 4) { // Complete
        if (req.status == 200) { // OK response
            responseStr = req.responseXML;
            var emailnumber=req.responseXML.getElementsByTagName("set").length;
            document.getElementById("chartframe").src="bargraph.htm";
        } else {
            alert("Problem: " + req.statusText);
        }
    }
}

Listing 4.1 code for Ajax implementation

In our implementation, the keyword search tool and the frequency analysis tool are implemented with the Ajax idea. For example, when the user submit keywords list, time intervals and interval number in frequency analysis tool, the JavaScript code behind instantiates the XMLHttpRequest object appropriately depending on browser, and then opens a request to the supplied URL. When the state of the request changes, the processStateChange() function will be called (which is going to display the bar chart with
The Google Web Toolkit (GWT) is a Java development framework that lets you escape the matrix of technologies that make writing AJAX applications so difficult and error prone. With GWT, you can develop and debug AJAX applications in the Java language using the Java development tools of your choice. When you deploy your application to production, the GWT compiler can translate your Java application to browser-compliant JavaScript and HTML. In our implementation, we use Google Web Toolkit to translate the main panels of topic map interface which is developed in Java first and then manually add dynamic features according to our request.

In the Email Frequency Analysis tool bar graph, the Date Period fields are viewed by the JavaScript for the page as buttons that bring up a new window containing the list of emails indicated by the associated bar. The Web Interface has been implemented using Struts.

4.2.2 Apache Struts

Apache Struts [35] is a Model-View-Controller implementation that uses servlets and JavaServer Pages (JSP) technology. Current web applications differ from conventional websites in that current web applications can create a dynamic response. Many websites deliver only static pages. A web application can interact with databases and business logic engines to customize a response. Web applications based on JavaServer Pages sometimes commingle database code, page design code, and control flow code. In practice, we find that unless these concerns are separated, larger applications become difficult to maintain. One way to separate concerns in a software application is to use a Model-View-Controller (MVC) architecture. The Model represents the business or database code, the View represents the page design code, and the Controller represents the navigational code. The Struts framework is designed to help developers create web applications that utilize a MVC architecture. In our
implementation, we are running struts on Apache Tomcat servlet container.

- **Struts Model Component**

  The Struts framework doesn't provide much in the way of the model, but it can integrate with any current model approach, including JDBC and EJB, as well as third-party packages like Hibernate, iBATIS, and Object Relational Bridge. One of the most important best practices for the Struts framework is that business logic should stay out of the Action class, regardless of what you're using for the Model layer. In our implementation, for example, for the frequency analysis request, the model component is the `doFrequencyAnalysis` class.

- **Struts View Component**

  The Struts presentation layer is almost as flexible as the model layer. You can use JavaServer Pages, HTML forms. But you can also use one of many alternative presentation technologies if JSP isn't your thing. From Velocity Templates to XSLT to JavaServer Faces, Struts easily integrates with many. In our implementation, the view component is the HTML forms with JavaScripts which we discussed in Section 4.2.

- **Struts Controller Component**

  When a request is sent to a Struts application, it is handled by the Struts ActionServlet. The Struts framework includes a concrete ActionServlet that for many users is adequate and requires no customization or additional work. When the ActionServlet receives a request, it inspects the URL and based on the Struts configuration files (see Listing 4.2), it delegates the handling of the request to an Action class. In our implementation, as you can see from Listing 4.3, the `frequencyAnalysisAction` class is part of the controller and is responsible for
communicating with *dofrequentanalysis* class in the model layer.

```xml
<action-mappings>
  <action path="/frequencyAnalysisSubmit" type="xhrstruts.frequencyAnalysisAction"/>
</action>
```

Listing 4.2. part of struts-config.xml

### 4.2.3 Main classes

Below is the list of main classes sitting on the server.

<table>
<thead>
<tr>
<th>Class Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>doFrequencyAnalysis</td>
<td>Implement the frequency analysis details and return the results.</td>
</tr>
<tr>
<td>FrequencyAnalysisAction</td>
<td>Parse the request of doing a frequency analysis, call doFrequencyAnalysis to do execute the analysis.</td>
</tr>
<tr>
<td>searchAction</td>
<td>Parse the keyword search request and call IRagent to get the search results.</td>
</tr>
<tr>
<td>IRagent</td>
<td>Implement the vector space based search.</td>
</tr>
</tbody>
</table>

Table 1 Main classes
public class frequencyAnalysisAction extends Action {

    public ActionForward execute(ActionMapping mapping,
                               ActionForm inForm,
                               HttpServletRequest request,
                               HttpServletResponse response) throws Exception {

        String csv = (String)request.getParameter("csv");
        csv = new URLCodecQ.decode(csv);

        StringTokenizer st = new StringTokenizer(csv, ";");
        String queryTerm = st.nextToken();
        String startDate = st.nextToken();
        String endDate = st.nextToken();
        String intervalNum = st.nextToken();
        StringTokenizer st2 = new StringTokenizer(queryTerm, ",");
        String[] queryTermList = new String[st2.countTokens()];
        int tokenNum = st2.countTokens();
        for(int i=0; i<tokenNum; i++){
            queryTermList[i]=st2.nextToken();
            System.out.println(queryTermList[i]);
        }

        String resp = doFrequencyAnalysis.startup(queryTermList, startDate, endDate, Integer.parseInt(intervalNum));
        System.out.println(resp);
        response.setContentType("text/xml");
        response.setHeader("Cache-Control", "no-cache");
        PrintWriter out = response.getWriter();
        out.write(resp);
        out.flush();

        return null;
    }
}

Listing 4.3. Code for frequencyAnalysis class
5 CONCLUSION AND FUTURE WORK

A model of a distributed data warehouse using a topic map to access knowledge from an organization’s collective memory has been presented. An initial version of the model has been implemented using Java as the programming language, GWT and Ajax to change the java code to topic map web interface, SOAP to communicate to data sources external to systems.

Future work may focus on the following aspects:

- Develop tools to integrate existing software tools associate.
- Implement user login system and user profile to make the topic system customizable at user level.
- Develop tools to allow users to save or download the results as images returned by tools.
- Solve the cross browser issue of the topic map web interface.
6 REFERENCES


International Conference on System Sciences Task Force on Organizational Memory


