An Enriched Web Services Client Architecture for Management and Sharing of Context

Bo Zhao

(Student id: 3084058)

(Email:zhaoboly@hotmail.com)

Supervised by:

Dr. Liam Peyton

Dr. Timothy C. Lethbridge

Thesis proposal submitted

To the Faculty of Graduate and Postdoctoral studies

In partial fulfillment of the requirement for the degree of

Master of Science

Under the auspices of the System Science Program

University of Ottawa

Ottawa, Ontario, Canada

May 2005

Abstract

The rapid growth of the Internet has brought us endless choice, but it is also difficult to find items according to personal interests.

Nowadays, a website is more like an island. In order to find useful information, you need to jump from one page to another page through the links. A web based system for managing context would give users the ability to manage personal context, collect interesting links, and attach notes, so that only one or two clicks would be required to reach the useful information. Such context is valuable, so it would also be useful if it could be shared with others.

We survey the technology available to create the Enriched Web Services Architecture in order to give a rich user experience in a browser application. We also survey current approaches and issues in managing and sharing context. Based on both surveys we propose architecture for context management and sharing. An example system, for Management and Sharing of Context (MSC) is built as a case study and evaluated. The example system helps people to collect links and articles in order to manage their context and share context with others.

Table of Contents

ABSTRACT **2**

1. INTRODUCTION **7**

1.1 Problem Statement 7

1.2 Motivation and Objectives 7

1.4 Thesis Contribution 9

1.5 Organization of Thesis 10

2. BACKGROUND AND RELATED WORK **12**

2.1 DHTML 12

2.2 Web Services 14

2.3 XML Topic Maps 16

2.4 eXtensible Access Control Markup Language (XACML) 17

3. ENRICHED WEB SERVICES CLIENT **19**

3.1 N-tiered Application Architecture 21

3.1.1 Distributed Systems, Client/Server Architecture and Web Applications Architecture *21*

3.1.2 A Three-Layer Architecture *22*

3.2 Reduction in Internet traffic 24

3.4 Facilitating development and deployment 27

4. MANAGEMENT AND SHARING OF CONTEXT **29**

4.1 Document oriented Web Services 29

4.2 Sharing Problem 31

4.3 Topic Maps in Management and Sharing of context 34

4.3.1 Design of the Topic Maps network *34*

4.3.2 Topic Maps Manipulation *36*

4.4 Privacy Implementation 38

5. CASE STUDY **41**

5.1 Overview 41

5.2 User Interface 42

5.2 System Architecture 44

5.4 Context Management 46

5.5 Context Sharing 52

5.6 Key Mechanisms for Dynamic Updating of the UI 54

5.6.1 Structural Cut and Paste *54*

5.6.2 Drag and Drop *55*

5.6.3 Asynchronous Server Interaction *56*

Deleted: 42

Deleted: 42

Deleted: 43

Deleted: 45

Deleted: 47

Deleted: 53

Deleted: 55

Deleted: 55

Deleted: 56

Deleted: 57

6. ANALYSIS OF THE RESULTS **58**
6.1 OVERVIEW 58
6.2 TEST SCENARIOS 59
6.2.1 Features and Functionality Comparison *61*
6.2.2 Total Request Count to the Server *62*
6.2.3 Total Response Time *64*
6.2.4 Total Amount of Received Data *66*
7. CONCLUSION AND FUTURE WORK **67**
7.1 SUMMARY 67
7.2 CONCLUSIONS 68
7.3 ISSUES AND FUTURE WORK 68
8. REFERENCES **70**

Deleted: 59
Deleted: 59
Deleted: 60
Deleted: 62
Deleted: 63
Deleted: 65
Deleted: 67
Deleted: 68
Deleted: 68
Deleted: 69
Deleted: 69
Deleted: 71

List of Figures

Figure 1 : The Traditional Web Page 13
Figure 2 : DHTML with Web Services support 14
Figure 3: XACML process [42] 19
Figure 4: Evolution of Enterprise Computing [40] 20
Figure 5: Change 1 layer to 3 layers 23
Figure 6: Transform XML to different formats 24
Figure 7: Enriched Web Services Architecture 25
Figure 8: Aggregate the new XML with the existing XML document 26
Figure 9: XML standard process 30
Figure 10 Public and private area in the server 31
Figure 11 Structure of a content object 33
Figure 12 : Topics and Associations 36
Figure 13: Customized Access Control 41
Figure 14 : the typical User Interface 43
Figure 15 : the Content Window 44
Figure 16: MSC System Architecture. 45
Figure 17 : Popup menu 47
Figure 18 : New Command 48
Figure 19 : Property Window 49
Figure 20 : Open a link content 49
Figure 21 : Open a file content 50
Figure 22 : Open a note content 51
Figure 23 : Drag and Drop 51
Figure 24 : Set the content to be public 52
Figure 25 : Share the content node 52
Figure 26 : Copy the content node 53
Figure 27 : Paste the content node 53
Figure 28 : Paste type dialog 54
Figure 29 : Asynchronous Server Interaction 57
Figure 30 : a simple context management system 59
Figure 31 : Total request count to the server for each interaction step (Y axis) 63
Figure 32 : Total Request Count to the Server 64
Figure 33 : Total Response Time in Million seconds (Y axis), for each interaction step (X axis) 65
Figure 34 : The Total of the Response Time 65
Figure 35 : Total Amount of Received Data in KB (Y axis), for each interaction step (X axis) 66
Figure 36 : The Total of the Received Data 67

Deleted: 44
Deleted: 45
Deleted: 46
Deleted: 48
Deleted: 49
Deleted: 50
Deleted: 50
Deleted: 51
Deleted: 52
Deleted: 52
Deleted: 53
Deleted: 53
Deleted: 54
Deleted: 54
Deleted: 55
Deleted: 58
Deleted: 60
Deleted: 64
Deleted: 65
Deleted: 66
Deleted: 66
Deleted: 67
Deleted: 68

List of Tables

Table 1 : user's entire set of activities in the system 60
Table 2 : Feature Comparison..... 61

Deleted: 61

Deleted: 62

1. Introduction

1.1 Problem Statement

After the introduction of the first corporate website in the 1990s, web content has increased dramatically. The Internet has become the most important tool for people's study, work or entertainment. Right now users are exposed to huge amounts of Internet content everyday. People need to remember a lot of website links in order to get valuable information. This is a big problem when you try to recall a useful web link, which you saw a few days ago, a few months ago or even several years ago. Companies have struggled to find the most efficient and cost-effective ways to maintain web content as a part of "corporate knowledge".[49] People are concerned with the following issues:

- How can we reduce the time delays in publishing the new content?
- How can we share the content while respecting privacy?
- How can we build up the context about web resources and maintain personal or corporate knowledge?

Nowadays, a website is more like an island. In order to find useful information, you need to jump from one page to another page through the links. You will not know what will happen next, just follow the link. Information is often not useful alone. Instead of finding the web page in isolation and out of context, people need the context to make sense of the information in a particular situation. We need to consider the information not only individually, but also within a context of other information from world wide resources. Contexts are in fact represented by metadata. Defining structured metadata embedded within the information will be used for organizing information.[51] Contexts will help us to better understand the information, shape the objectives and make decisions.

1.2 Motivation and Objectives

Users are exposed to a large number of website links everyday. These links are often too long to be remembered by people. All widely-available browsers provide a 'favorites' or 'bookmarks' function to help the user save and organize website links. But the browser

saves these links in the user's own file system. That means if you use another computer, you will lose all your favorites/bookmarks information. Also transferring your favorites/bookmarks information to a different computer could be difficult for normal users.

The benefits of storing the user's favorites in a central server, and adding specialized software to manage the links, are listed below:

1. All metadata information for the favorites can be saved, including category.

2. Greater portability and flexibility of retrieval from different computers.

3. Greater opportunities for bookmark searching.

4. Ability to have a customized category tree to make the best use of your screen and a completely configurable display of expandable folders. A user can quickly navigate to any pre-saved web resource and add, delete or rearrange category based on the user's organizational pattern.

5. Ability to allow users to easily add any information about links, rank the sites they visit and share the information with other users

All of these above advanced features are not supported by the browser. There are some products for managing bookmarks on-line:

- Yahoo Bookmarks. Bookmark favorite web site online and access bookmarks from any PCs. [52]
- Spurl.Net. The user can search previously saved links, notes and even the entire text of all the pages which have be saved. The user can discover interesting links through recommendations, hot lists and more.[53]
- MyBookmarks. Their full featured editor makes it easy to organize and search bookmarks online. It also allows users to make bookmarks public so the user can share them with others.[54]
- Backflip. Save pages into personal directory. Share web sites with friends or publish them on the web.[55]

But they are all based on the traditional web page and provide limited flexibility and customizability to make their client applications better.

The objective of this study is to help people better understand and make use of available information around the world wide resource in the best possible way. Hopefully, it will support the users to perform their regular work tasks. It will provide a framework that allows user to retrieve and reuse the information that is available on the Internet. It also helps the company better organize corporate knowledge.

Our system will allow a user to organize and share the Internet resources using web-based controls in a zero footprint web browser. The user can build up a personal context-centric view of the Internet resource by creating customized content hierarchy. The group manager can combine personal hierarchies into a group content hierarchy.

This thesis investigates the following issues:

1. Can a rich enough interaction be provided in a web browser to support context management by leveraging DHTML?
2. Can an efficient web architecture be put in place to support the browser interaction using web services?
3. Can this infrastructure be extended to support context sharing between users in a safe and controlled manner?

1.4 Thesis Contribution

In this thesis we will develop an architecture that will make efficient use of web resources to support personal context managing and sharing.

The main accomplishment of this thesis is to design an approach to create fat web services client incorporating business logic, which has zero footprint. This was accomplished using JavaScript operating in the browser. It will leverage DHTML to provide an interface that enables more functions and convenient interactions than are possible using a traditional web interface.

This thesis is an investigation of effective methods based on emerging Internet Standards to manage web context and share that context with others. The open standard XML Topic Maps for organizing content will be investigated as well as standards for

accessing content via Web Services (SOAP, WSDL, and UDDI) and for securing access to content (XACML).

These contributions result in the Enriched Web Services Architecture which produces benefits at the user level, system level and implementation level.

At the user level, the Enriched Web Services Architecture provides richer user interaction, more features and more usability. The fat web services client will support cut and paste, tree view, tab window, drag and drop, popup menu and context sharing while respecting the privacy.

At the system level, the Enriched Web Services Architecture provides standards-based approach to organizing private and publicly shared context. It provides centralized standards-based security and needs less administration efforts to install clients.

At the implementation level, the Enriched Web Services Architecture results in less network traffic, quicker response in the user interface, standards based implementation.

1.5 Organization of Thesis

The rest of the thesis is organized as follows:

Chapter 2 provides some background on DHTML, Web Services, Topic Maps and XACML.

Chapter 3 discusses the abstract architecture of Enrich Web Services Client and its benefits.

Chapter 4 discusses technologies used for management and sharing of context. These technologies include Web Services, Topic Maps and XACML.

Chapter 5 is a case study in which a system for Management and Sharing of Context (MSC) is built based on the previous chapter's discussion.

Chapter 6 outlines the test scenario and discusses the results of the experiment.

Chapter 7 summarizes our thesis and gives direction for the future work.

2. Background and Related Work

The growth of the Internet has been explosive during the last decade. The World Wide Web has become an information resource for many people's work and entertainment.

With the rapid growth of the Internet, users are exposed to endless website links everyday. In order to efficiently use Internet resources, people need an integrated software solution that can manage the distribution of web content in a simple application. An integrated solution will increase the quality and relevance of web content. [50] It is also important to find out the related information and recommendations from the help of others.

Such an integrated software solution will help users build an individual specific hierarchy to define a personal context view of web resources. While doing that, users are building up the relationships between items on the Internet. Increased relevance leads to reduction in cost of reacquiring information. When building up a personal context view of the Internet links and sharing it, user's looking to educate them on a new subject area can do so more efficiently. The company can maintain the corporate knowledge by combining all the personal context views.

2.1 DHTML

"In a traditional web page, a user would typically select a link on the page presented in the browser, which would lead to another HTTP request."[47] The client browser navigates to a new URL, renders the HTTP response and displays the entire page. This method is used today but it is inefficient. Even to change a small part of a web page, the entire web page will be refreshed.

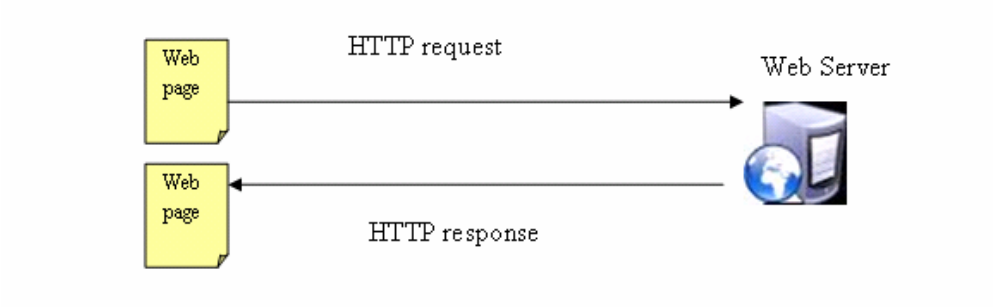


Figure 1 : The Traditional Web Page

"DHTML is a combination of technologies used to create dynamic web site" [43]. The basic idea of DHTML is to allow the tag of a web page to be changed at any time and make the web page more dynamic. It is a combination of four components: HTML, CSS (Cascading Style Sheets), Scripting and DOM (Document Object Model).

- HTML: HTML defines the basic web page structure using tags such as body, forms and tables.
- CSS (Cascading Style Sheets): CSS define how to display the HTML elements. Using CSS, we can control the font size, link color, background and many other attributes of our web page.
- Scripting: Scripting in the client side provides the scripting language to manipulate the web page according to the user's actions. Scripting adds dynamic and interactive behaviour to the web page.
- DOM (Document Object Model): It is the standard of W3C [14]. It provides a more natural way of look at the XML document and HTML. Using a programming language, a XML document can be parsed into a DOM object. Walking through the DOM tree, the elements and their attributes can be modified, deleted, searched and created.

From Internet Explorer 5.0, Microsoft has provided Web Services behavior. It enables a client-side script to invoke remote Web Services. This component enables Internet Explorer to retrieve data from remote Web Services and to update a web page dynamically using DHTML. "The WebService Behavior is implemented with an

HTML Component (HTC) file as an attached behaviour." [46] It provides an easy way to use Web Services in the browser. [46]

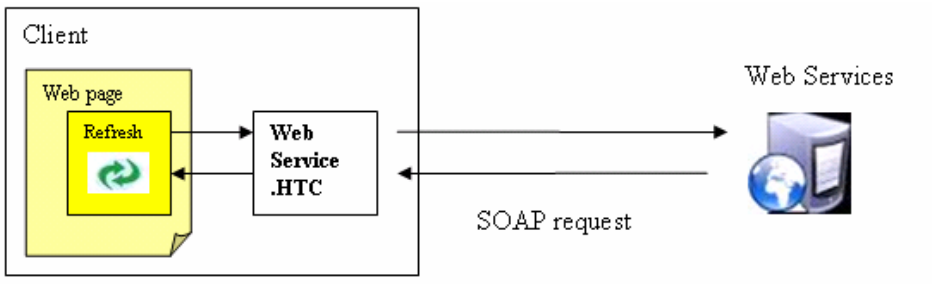


Figure 2 : DHTML with Web Services support

DHTML can be used to build an Enriched Web Services Client with more features and actions possible than the plain HTML. DHTML uses a scripting language to change a web page into a DOM object. When receiving a method call from the client, the client-side scripting language sends a request to the Web Service using WebService behaviour. From Internet Explorer 5.0, Microsoft has provided an HTML Component (HTC) file as an attached behaviour." [46] It provides an easy way to use Web Services in the browser. [46]

The results of the Web Service request are returned to the client script. The client script makes updates to the DOM object in the browser, and only partially redraws the web page to reflect those changes. That gives a rich user experience in a browser application.

2.2 Web Services

In order to communicate with each other, we need a common language to ensure that the content has the same meaning to everyone using it. Two software agents should exchange information and understand each other even if they use different "vocabularies" (different programming language). The aim of Web Services is to provide a standard framework to enable application-to-application communication.

Nowadays, the Internet maintains its *content* in a distributed manner and every piece of content is maintained by the different website. Web Services provides an architecture in which *function* is distributed. We can consider Web Services as reusable software components across the Internet; they form key building blocks.

"Web services are self contained, self describing, modular applications that can be published, located, and invoked across the web. Web services perform functions which can be anything from simple requests to complicated business processes." [22]

Web Services is a new service-oriented infrastructure [23]. According to the previous definition [22], the services are simply executable functions located on the Internet. From very simple functions to complicated business processes, every process can be a service. They can be used alone or combined with other services to carry out complex business processes or tasks.

In the view of the corporation, reusable services can be distributed into different servers on different platforms and then can be ready to be consumed by clients or other applications. Also, within the rapidly changed IT area, new Web Services can be continuously added to the existing Web Services. This will provide a perfect dynamic, distributed architecture [24].

The Web Services infrastructure includes three basic standards: First, Simple Object Access Protocol (SOAP) is a XML based protocol to exchange the information in a distributed environment. Second, Web Services Description Language (WSDL) defines how operations can be invoked using protocol binding. Finally, Universal Description, Discovery and Integration (UDDI) provides the platform for the clients to find the other Web Services.

SOAP (Simple Object Access Protocol) is a simple XML based protocol to let applications exchange information over many different protocols including RMI, HTTP, SMTP. SOAP has three key attributes of great importance:

First, SOAP is a *communication protocol*. It is designed to allow Internet communication between programs. Using SOAP, you can integrate useful data and tools from various websites. For example, you can include the Internet query service from Google in your application.

Second, SOAP is *platform and language independent*. Web Services define a standard lightweight infrastructure to allow different applications on different platforms to communicate with each other.

Third, *SOAP is based on XML* (eXtensible Markup Language), which is simple, well structured and extensible. SOAP simply sends a XML envelope and returns a XML envelope from the server. XML allows the author to define their own document structure. Everyone can understand your document along with a DTD (Document Type Definition) or XML Schema. XML is the future for data manipulation and transmission over the Web.

Microsoft, Sun Microsystems, IBM, Oracle and several others have developed tools for building Web Services.

2.3 XML Topic Maps

Topic Maps is an open standard for organizing digital documents. In theory, Topic Maps can be used to represent any real world subjects and relations. The design goals for XML Topic Maps can be found at its homepage page [38].

Topic Maps are designed to build the semantic information network over any kind of information resources. Topic Maps are an XML document in which different elements are used to represent topics, occurrence of topics, and associations between topics.

The key concepts of Topic Maps are [38]:

- Topic Type: A Topic Type is like the class in object-oriented programming. For example, “Country” can be a Topic Type.
- Topics: A topic is an instance of the Topic Type. A topic can be anything that represents the real world subject. For example, the “Canada” Topic will be an instance of the “Country” Topic Type. Each topic should have one or more topic type. The relation between topic and topic type is typical class-instance relation. A topic may have a base name, display name and sort name.
- Associations: The association connects two topics and provides a semantically correct meaning. Every association has an association type, which is defined as a topic in XML Topic Maps. The relation between association type and association is also a typical class-instance relation. For

example, we can define a “is to the north of” association to connect two topics: “Canada” Topic and “USA” Topic. This association means that Canada is to the north of USA.

- Occurrence: A topic can have occurrences, which are information resources related to the topic. Normally, an occurrence will be an URI. For example, “Canada” Topic will have an occurrence: http://canada.gc.ca/main_e.html.

2.4 eXtensible Access Control Markup Language (XACML)

The Organization for the Advancement of Structured Information Standards (OASIS) is a non-profit organization responsible for developing worldwide standards for e-business.[34] The eXtensible Access Control Markup Language (XACML) version 1.0[36] has been an OASIS standard since 2003. New improvements were made in the draft version 2.0[37]. XACML was developed by a team that included people from Entrust Inc., IBM, OpenNetworks.org, Quadrasis Inc., Sterling Commerce Inc., Sun and BEA Systems Inc.

The benefits of XACML are listed below [35] [28]:

1. XACML provides one standard access control policy language and the policies need not to be written in many different languages.

2. XACML policy is flexible and extensible. User can use any kind of properties to define the policies and can express different conditions on context data.

3. XACML allows one policy to refer to another.

4. XACML is distributed. The XACML control point can be distributed across several servers and the policy can be managed by the different groups. XACML defines the standard way to combine the results from different policies into one decision.

XACML is an XML-based standard that contains an access control policy language and a request/response language. “XACML is better suited to be applied in industry as an access control language for, but not only, XML documents.”[25] But the language specification and the policies are much more complex than other access control language such as XACL and Fine-Grained Access Control [25].

A typical scenario of using XACML is that a user wants to take some actions on a particular resource within a certain situation. In this scenario will be a logged-in user who wants to access a content object. The user will log in and retrieve personal account information. When a user wants to access to a content object, the user interface will make a request to the service which protects the content resource. This service is called a Policy Enforcement Point (PEP). The PEP will encapsulate the request with requester's attributes, the requested action and other environment information. The PEP then sends this encapsulated request to the Policy Decision Point (PDP), which will evaluate the request and call the policy which is applied to the request. The PDP then sends back a response about whether the access should be granted.

The user from previous scenario is referred as a *Subject* in XACML. In this thesis, the user should be a logged-in user and the user is identified by a unique string. A Subject is an actor who has specified attributes that will help apply the policy by the PDP. The content object is referred to as the *Resource* and the action is called an *Action* in XACML. The Action declares the access mode that the Subject can use on the Resource. In this paper, the actions may include: read, write, modify.

Once the PDP receives a request, it will look the particular policy or policy set that is applied to the request. A policy set will have multiple policies. A policy contains multiple rules and conditions. It is evaluated based on the request and the response can be: Permit, Deny or Indeterminate.

The typical XACML process [42] is shown in Figure 3.

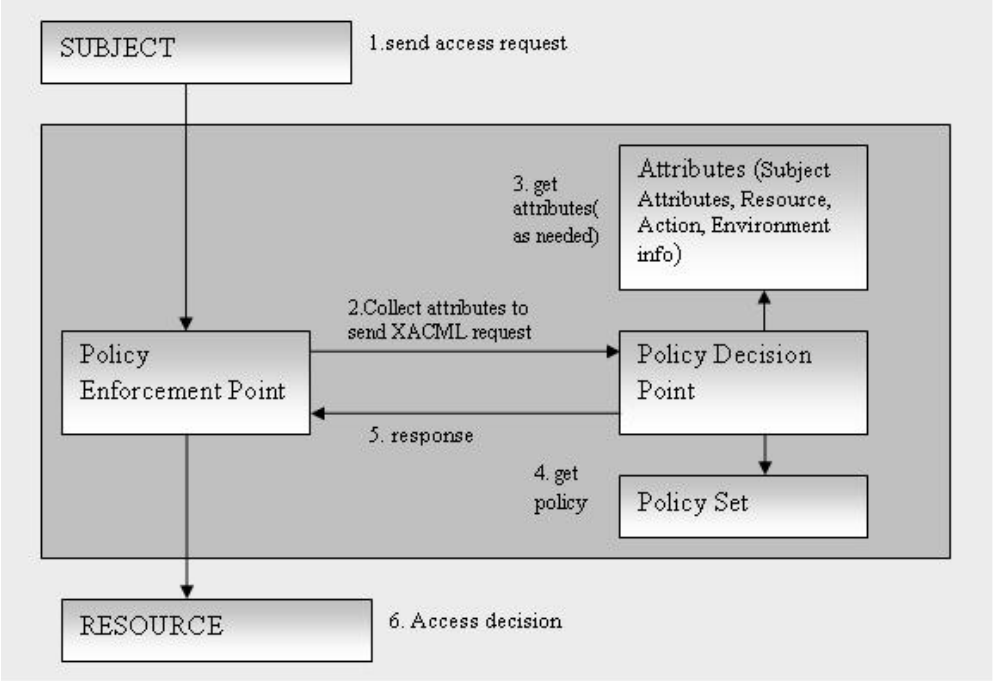


Figure 3: XACML process [42]

3. Enriched Web Services Client

The evolution of enterprise computing has been driven by innovations in system architecture [40]. In the mid 1970s, central mainframe computers held all application functions and enterprise data. Terminals provided a very simple user interface via a direct connection to the mainframe. With the introduction of personal computers, client/server architectures became popular; these enabled companies to distribute information and processing throughout the organization. The computing workload was divided between the personal computer client and the server. Right now, web servers allows companies to distribute data to more and more end users, which today are using relatively simple user interfaces in a browser. With the Enriched Web Services Client, companies will again be able to distribute information and processing in a more intelligent manner. Figure 4 shows the evolution from a centralized mainframe computing model towards a distributed web services model.

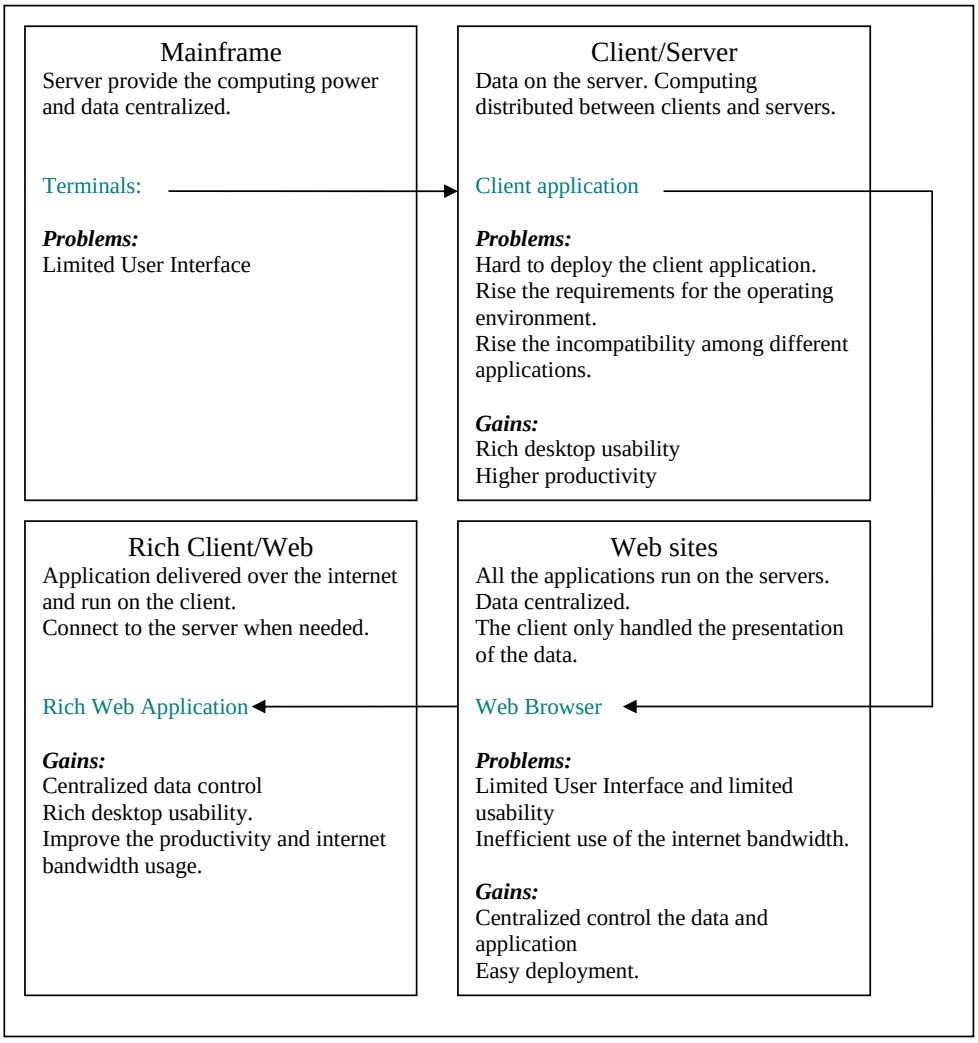


Figure 4: Evolution of Enterprise Computing [40]

3.1 N-tiered Application Architecture

3.1.1 Distributed Systems, Client/Server Architecture and Web Applications Architecture

A distributed system consists of a collection of autonomous computers, which are connected through a network. This system enables computers to coordinate their activities and share the resources of the system, so that users perceive the system as a single, integrated computing facility. Distributed architectures can be used for web applications.

The web application architecture can be divided into two types: the Thin client architecture and the Fat client architecture. [44] Both have client side components and server side components. The difference is concerned with where applications and components execute and reside.

- FAT client. The Fat client has Business Logic layer and requires a large footprint as software needs to be installed on the client and there is often local configuration of the machine required as well as local storage of data.. The server hosts the database and processes data requests from the client.
- Thin client. In the thin client system, the client is made as small as possible. The client is zero-footprint and has no Business Logic Layer. It will only include Presentation Layer. The typical Thin Client is Browser application which includes HTML and small amounts of JavaScript.

Footprint refers to the quantity of resources (applications, OS extensions or patches, data, middleware, extra hardware, extra memory, extra network protocols, etc.) that a user has to install on his/her computer prior to using an application or service. Footprint does not include resources that come with hardware and operating system configurations that have been widely available in the recent past and present, so requirement for a browser does not increase footprint unless only a specific browser is supported. In the zero-footprint web application, only JavaScript and HTML are sent to the client. So the web application can be easy to redeploy and manage.

N-tiered architecture divides the application into several tiers, so that each tier can be developed, changed and maintained independently. Almost all web-based UI applications have an N-tier architecture, which can be divided into: Presentation tier, Business Logic tier, Data Access tier and Data tier.

Traditionally the web application is thin-client Client/Server architecture. On the client side, it only includes HTML/XML pages for presentation purposes. HTML user interfaces are simpler to use and easier to learn than earlier generations of client-server systems.[48] However it is too simple for certain types of applications; and application developers need greater flexibility and customizability to make their client applications better.[44] .

In our work we wish to create an Enriched Web Services Client, which is a fat client with zero-footprint by leveraging JavaScript and XML in the browser to create a business logic layer.

3.1.2 A Three-Layer Architecture

An important contribution of the rich client architecture is that we change the one-layer browser client into a three-layer client. Each layer can be developed and deployed independently. Figure 5 shows the following three layers:

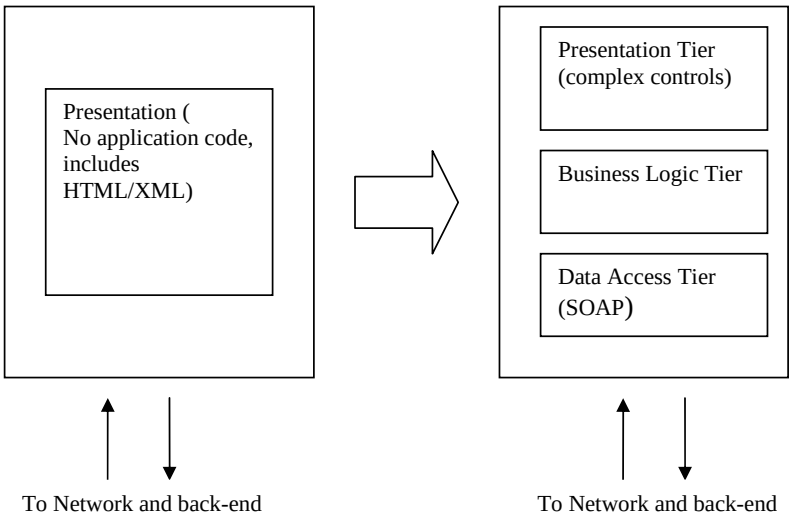


Figure 5: Change 1 layer to 3 layers

Presentation tier: This tier handles the user interface, and provides the user interface with the look, feel and behaviour of a modern desktop application. The user interface includes tree view, grid, graph controls, text area, etc.

Business Logic tier: The Business Logic tier contains all the knowledge required to modify the data model. [45] This tier renders XML, and delivers it to the Presentation tier. It holds the XML Data Model which is used for different presentation controls. This tier also includes data verification and data cleaning.

Data Access tier. This tier uses SOAP to fetch the data from the server when necessary.

Dividing the client side into three tiers, we can separate the content from the presentation. Using predefined templates, we can transform the XML document into all kinds of presentation formats such as tree views, property windows and ordinary web pages.

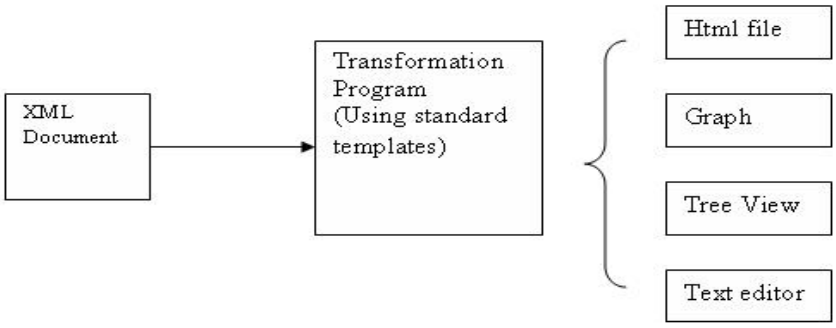


Figure 6: Transform XML to different formats

In this Enriched Web Services Client Architecture, the server is freed from much of the workload it traditionally does because the client side will perform all the presentation work and some business logic tasks. The client will not be limited to connecting with one server. The client application can access Microsoft, IBM, and Oracle databases over the Internet when needed. Compared to the traditional Thin Client model, the chief benefits of the Enriched Web Services Client Architecture include:

- Simplified application programming: Each layer can be developed independently, and can be connected to the existing applications and infrastructures.
- Easy deployment: No installation is required and it is easy to add extensive data manipulation functions.
- Improved performance: The client takes the presentation and some business logic workload from the server and executes it locally.

3.2 Reduction in Internet traffic

In a traditional web page, a user clicks on a URL link of the page, and the browser invokes another HTTP request and returns a completely new web page from the server.

Assuming the average size of a web page is 5-10 KB [32], every user's operation will invoke a HTTP request and have a 5-10 KB result.

The Enriched Web Services Client provides a more efficient and effective way to get Internet content. The end user can view Internet content in various ways. In the Traditional Web Page Architecture, all Internet content, bit by bit, will be transferred over the Internet. Currently, most of the websites use this model. On the other hand, in the Enriched Web Services Architecture, the client PC has a large number of choices of application functions and data sources. What is needed to be transferred over the Internet is a small amount of the dynamic data to add specified details of the content, which are not available in the client PC. Figure 7 shows an Enriched Web Services Architecture.

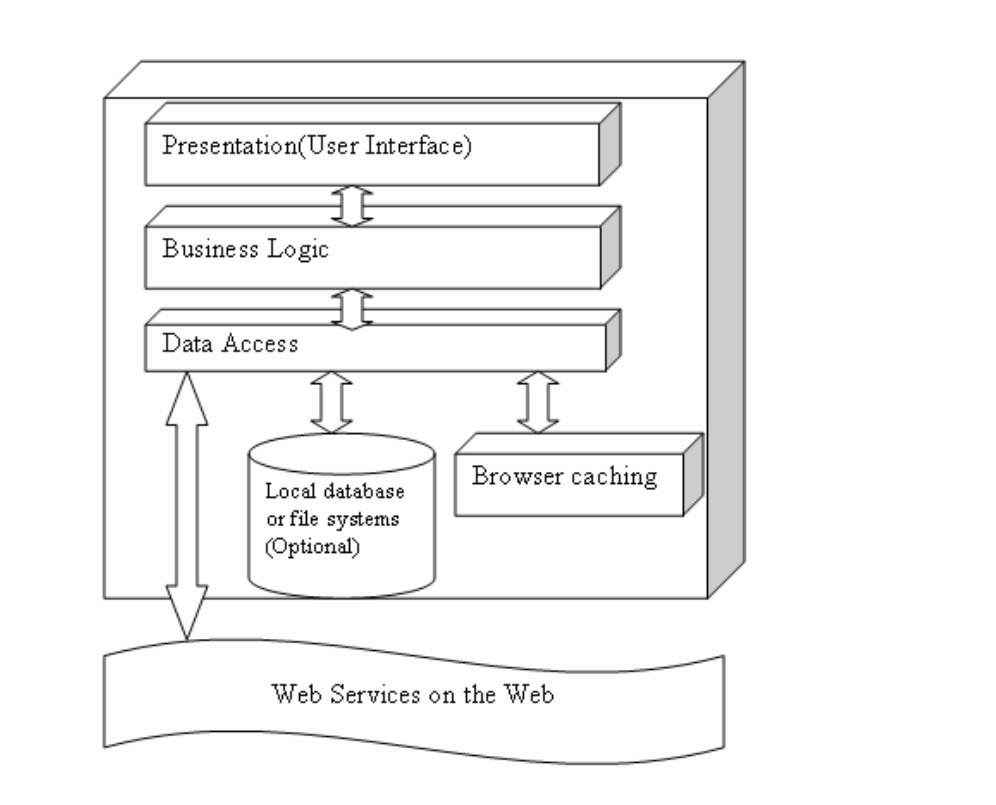


Figure 7: Enriched Web Services Architecture

The Enriched Web Services Architecture can make intelligent use of local PC storage to increase the efficiency with which it uses Internet bandwidth. Most requests

from the presentation tier can obtain needed results locally without communicating with a web server. For example, most conventional web application navigation uses URL links that invoke other HTML pages. Even walking through a site will frequently change the web page. The tree view navigation described in this thesis handles the operations locally and reduces the need to communicate with web server.

When surfing the web, a user continuously jumps from one page to another page and can become easily lost in the "web ocean". The architecture described in this thesis can also help the user organize their favourite websites in a tree view structure so these can be a starting point while surfing the "web ocean".

XML is a very flexible text format which has become the industry standard for content transformation and delivery. XML is generic approach to store and manipulate content [19]. When some operations can not be handled locally, the client will send a Web Services request and get an XML document as the result. We can aggregate the new result with the existing XML Data Model and next time need not get data from the server.

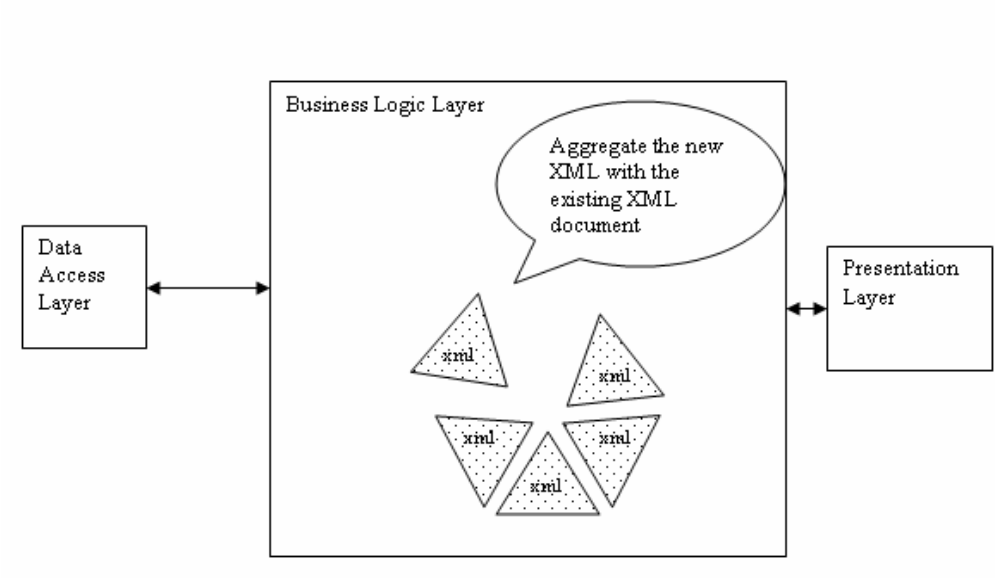


Figure 8: Aggregate the new XML with the existing XML document

Since most of data can be retrieved directly from the local machine, the user will see dramatically improved response time and reduced network activity. The client application

will run on the end user's local machine, not on the server. It will request the additional data from the back-end server only when needed.

The server is freed from most of processing work that it traditionally does and the network is freed from the constant back-and-forth of web page requests and sending fresh web pages.

3.4 Facilitating development and deployment

Many Windows programs require the running of a setup procedure that modifies the client's operating system files, such as the Registry, DLLs, and system variables. A lot of problems arise due to different versions of the operating system and incompatibility of different software libraries.

The Enriched Web Services Architecture can be built in the context of the web browser, so it has all the benefits of the web browser. The web is therefore an excellent vehicle for zero-footprint deployment. The server side machine is completely hidden from the user. All a user needs is a web browser, and nothing new needs to be installed. When a change or upgrade is required, it will only affect the server side. That will reduce the efforts needed by administrators to deploy and upgrade the application. They only have to upgrade the server and not thousands of client machines. (However, it is true that developers of the application are faced with significant challenges in ensuring that the application will run on different browsers from different vendors, or on different versions of the same browser. Administrators still have the burden of keeping browsers on all the thousands of client machines in synch with the applications that need to be supported).

There are of course challenges with rich applications hosted in a browser. Accessing the local file system is typically not allowed for security reasons although this would be useful to cache state information in the event that the network goes down. Although there are mechanisms to support local file access (including the use of HTA files in the browser) these are still dependent on users being sophisticated enough to change their settings (and for administrators to allow this behavior). Book marks are also typically problematic in an Enriched Web Services Client application. The original URL that was used to start the application has no relationship to the myriad of state changes that may have occurred in the client.

Currently, popular browsers include Internet Explorer (IE), Mozilla, Netscape. They provide completely different Web Services functionality. Both the server and the client can detect the vendor and version of the browser. The server can send a completely different set of HTML and scripting depending on the browser. On the client the scripting is interpreted so one can test for the browser version in order to adjust which code steps are taken.

4. Management and Sharing of Context

In this chapter, we will discuss issues and technology related to the management and sharing of context on the Web. This chapter is organized as follows:

Section 4.1 Introduce the Document Oriented Web Services and the benefits of this technology.

Section 4.2 Discusses issues related to the sharing of the context and briefly introduce how these problems can be handled.

Section 4.3 Apply Topic Maps to build up a context-aware structure for web resources. It includes: build a semantically meaningful network and Topic Maps manipulation.

Section 4.4 Introduce the concept of XACML. It discusses issues about how to use XACML to protect the privacy.

4.1 Document oriented Web Services

Web Services are frequently described as distributed object technology. That mainly comes from the name of the transport protocol: SOAP (Simple Object Access Protocol). However, it uses XML, but is not limited to it. Web Services actually have two different types: object-oriented and document-oriented.

Object-oriented Web Services are simply calls to remote methods. When the client sends a SOAP message, even the message is XML-based, it actually describes the remote object. The server receives the SOAP request and translates it into the programming object (e.g., a Java object, C# object). In this way, Web Services are a distributed object technology. The developer need not know the details of the SOAP message. Every object is directly mapped from the message automatically by the developing environment.

In document-oriented Web Services, the business XML document is delivered over the Internet. The document is not understood by the programming language, so it can not be mapped directly to the objects or method calls. Actually, they are the self-contained, self-describing business standard documents. This process is as if both client and server sign a contract to agree on a standard document structure. So, both will know what they are doing. Figure 9 shows the document-oriented Web Services process.

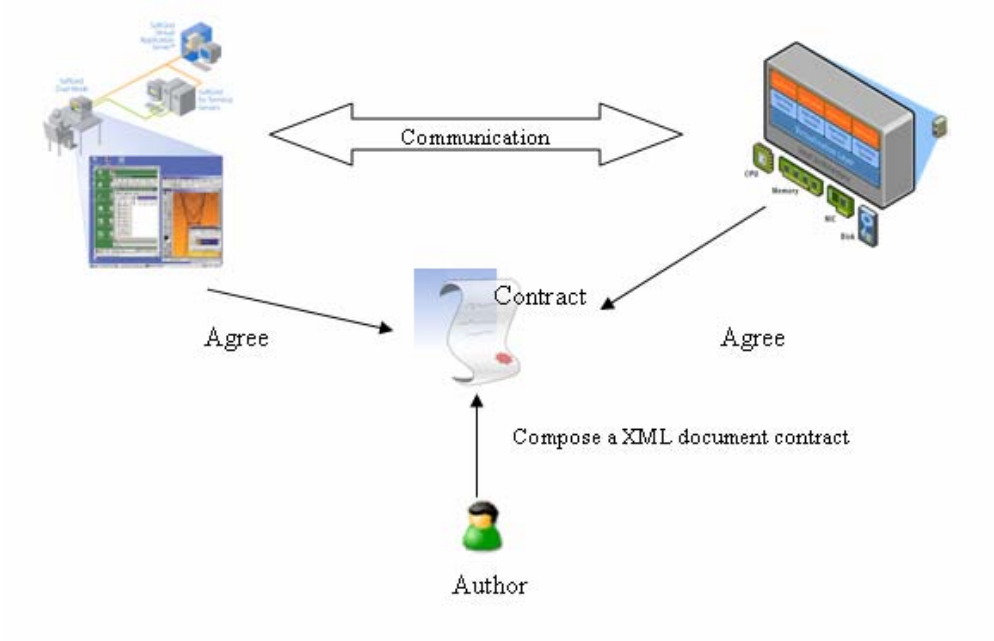


Figure 9: XML standard process

The good practice is to use document-oriented Web Services. Even if the object-oriented approach is implemented very quickly; the developer will have a lot of trouble in the future when he has to interact with other resources that are beyond his control.

The document-oriented approach has a number of benefits:

1. As a rich and self describing data structure, XML can contain much data and it is well organized. It will provide more information than simple parameters.
2. The XML document is separated from the implementation. The client and server development is totally independent. For example, the client side can be developed using C#, and the server side can be Java web server. Both can communicate based on the standard XML document. This standard XML document does not contains any implementation information. The client side application simply sends a request and waits for a response, and the server side can make a series of method calls or even call other Web Services in order to finish the task.

4.2 Sharing Problem

The user will log in and have a private space. In the private space, the user can organize the content object and modify attributes of the content object. In the public space, the user can view content objects that are published by the other users. These public content objects can be copied or linked into the user's private space in order to help build up personal knowledge assets. Figure 10 shows the public and private area in the server.

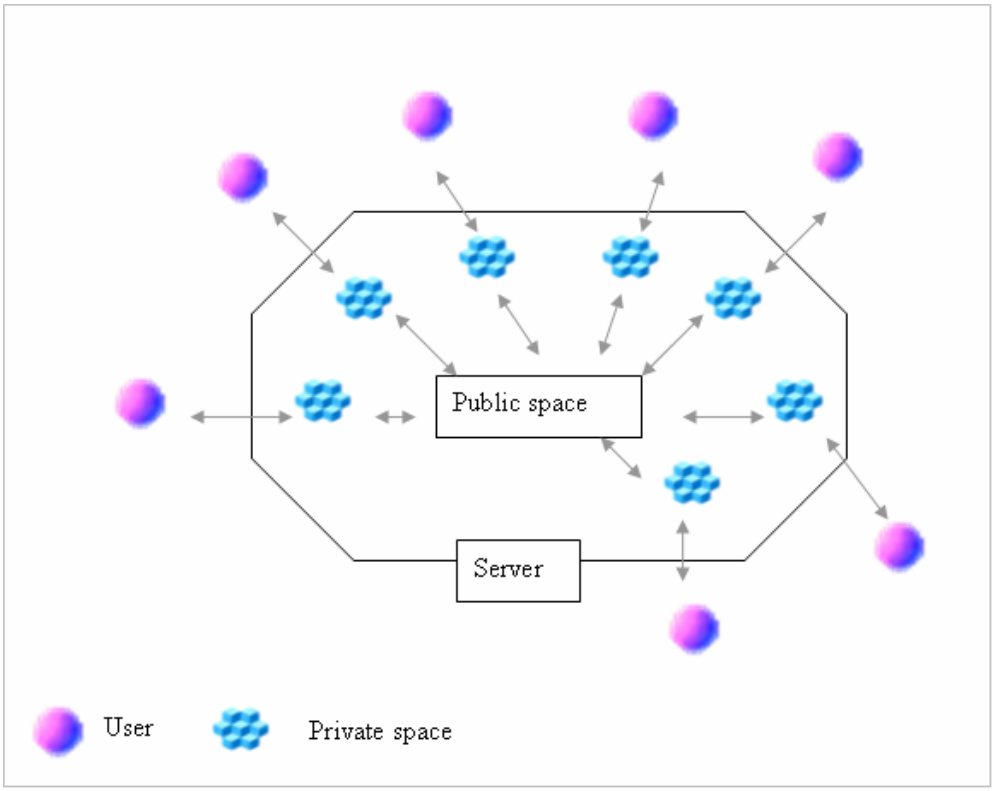
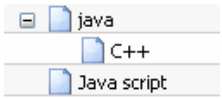


Figure 10 Public and private area in the server

Everything is for the user. Each user who comes to the site expects the site to work for him. He will not thinking about all of the other users. He will organize his own web content either from the empty template or from the existing resources. What he most focuses on is, "How and where do I get the valuable information that I need?"

Content is King. The most important thing is the content. All processes are about how to handle the content. Content must be accurate, informative and fresh.

In this thesis, the basic component is the content object. To simplify this, you can consider the content object to be a one element node in an XML document. In the following picture, there are three content objects. The content object is described as the topic in Topic Maps.



Every content object has the following parts:

- Content: This type of data is used to define the unique content. It can be a web link, a quick note stored in the database or a file.
- Metadata: Including some specific data added by the system. Such as: author name, creation date, display name, relationship to the other content object, etc.
- Template: Using which template to open, modify the content object.
- Access rights: What kinds of actions are allowed on the content object? This data will define the privacy level and help regulate who can do what in which specified situation. The policy rules can be predefined policy or individual customer-based policy.

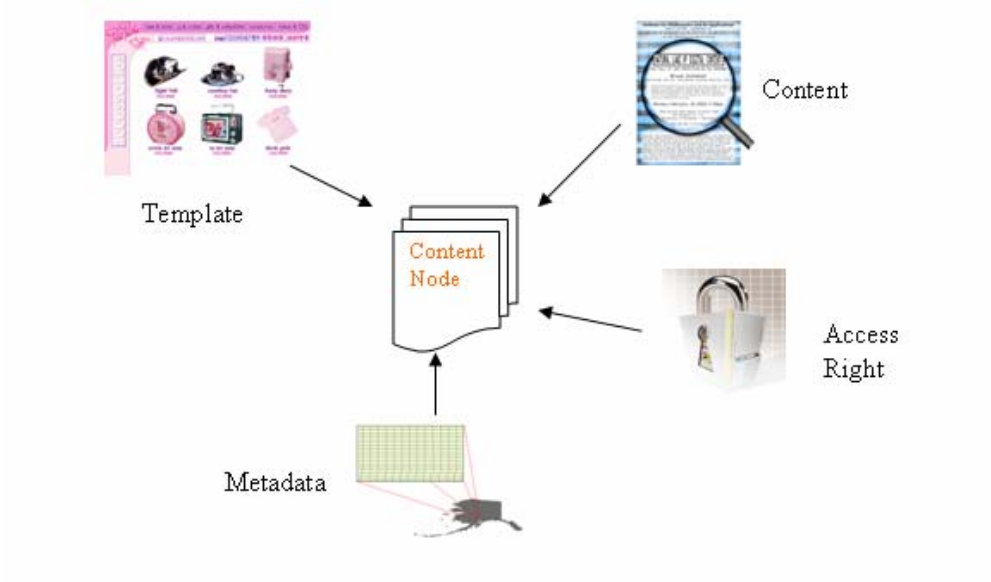


Figure 11 Structure of a content object

There are two possible ways to create a content object: Creating a new content object from the empty template or creating from an existing object.

For the first way, the user should select a template and type in the metadata of the object. In our case study, described in chapter 5, three types of templates were provided: uploaded file, quick note and web link. In the future, more templates should be provided. If the user selects the upload file template, the user should upload a file to the server and then fill in the metadata.

For the second way, the user can look through the public area which contains all content objects that have been shared by the other users. The user can drag and drop the content object into the current user's private area. The user can select to create a link or clone the content object. Only the cloned content object can be modified by the current user.

When the user tries to open or modify the content object, a specified template will be used and provides enough functions to process the data.

4.3 Topic Maps in Management and Sharing of context

4.3.1 Design of the Topic Maps network

Based on the previous scenarios, we can apply Topic Maps to build up a context-aware structure for web resources. Topic Maps provide appropriate concepts to build a semantic link network among document objects. In this approach, Topic Maps are applied to show the power and benefits of the Enriched Web Services Architecture. Organization and manipulation of a complex Topic Maps are hard to be done in traditional web pages.

A semantic network should be designed for management and sharing of all kinds of data. A topic will be the basic construct of the data and it can be inserted into this network as a network node. Each node can be connected to any other node as needed and these connections will provide semantically meaningful associations. A user navigates through the network by following the associations between the topics.

First we should define the topic class and the association class, and then every instance of the topic class will have zero or more instances of the association class. They are the base set of topics that are available to the users and users can build their own personal context view from them.

1. Topic Classes and Meta Data:

- Person: Since the system is designed mainly for personal usage, people should be a topic class for the implementation of this system. The basic metadata will include: name, title, e-mail, phone number and address.
- Web Link: This class defines the web resource. It will usually include a URL (Uniform Resource Locator). It can be any content from the Internet such as web page, XML, document, picture and so on.
- File: It is almost the same as the Web Link class. The only difference is that the URL will link to the file which the user has uploaded to the private area on the server.
- Note: This will be a quick note that a user creates on a web page. The content of the note will be saved in the database.

Web Link, File and Note have almost the same metadata such as: URL, title, date created, keyword and comment. Metadata of each kind of Topic class can be assigned by means of key-value-pairs. The key defines the type of Metadata and has to be unique within a topic.

2. Association class:

An association in Topic Maps defines two members as their endpoints. The user can navigate the network by following the association from one topic to the other. It also provides a semantically correct meaning of the relationship between the topics. It will enable the user to establish the context of the topic. The basic association classes will be:

- “Authored by” association: The first member will be an object of Person class and the second member can be any object of the three classes: Web Link, File and Note. It means that the current object is “authored by” this person.
- “Contains” association: This association connects any two objects of the three classes: Web Link, File and Note. It is a very general association which means that one topic contains some other topics.

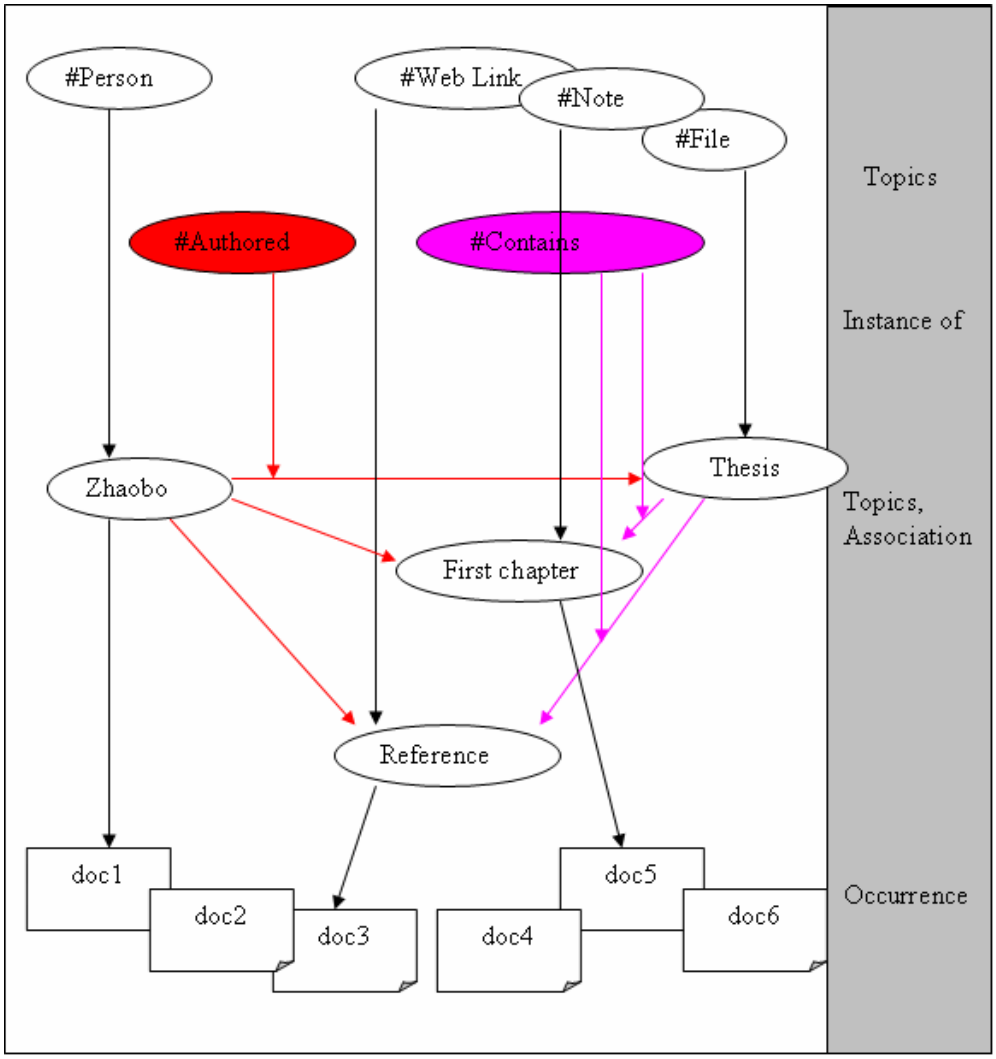


Figure 12 : Topics and Associations

4.3.2 Topic Maps Manipulation

This section will focus on the usage of the Topic Maps network as a means of context management. We discuss insertion and removal of topics and associations, navigation throughout the network and the merging of topics.

1. Enter the Network

We use a tree structure to display the Topic Maps, so first we should decide the entry points so that the user can start navigation from the entry points. Also, we use entry points to separate the user's private area from the public area.

We define three new Topic classes, and every user will have one instance for each of these three classes. These are the entry points when a user navigates the network.

- Private area: A user uses "Contains" association to connect to any other topic. The user is the only one who is able to read (and insert, delete …) the sub net.
- Public area: When a topic is set to be public, a new "Contains" association will be created and connected to this public area topic.
- Domain: The topics in this area are organized by the group organizer and include topics from other users. They are shared by all users.

From the entry point, we will use the "Contains" association to find out all the connected topics and display them in the tree.

2. Insertion

When a user wants to insert a new instance of Web Link class, File class or Note class, a new "Contains" association will be created to connect it to an already existing topic in private space. If no topic is specified, the default topic will be the current user's private area topic. Also, the "Authored" association will be created.

If an existing sub tree from other users is inserted, it simply creates a "Contains" association to the current topic.

3. Removal

Whenever a topic is deleted, all its associations to other topics have to be deleted as well. Also, all the subnet topics that are connected by the "Contains" association will be deleted.

4. Merging Topics

A Subject-based topic merging rule is covered in the XTM processing model 1.0 [39]. The process of this rule makes sure that whenever two topics represent the same subject, they are merged. Usually the URI (Uniform Resource Identifiers) of the resource is used to identify the subject. Two topics which have the same URI have to be merged into one topic. Every topic has different access control for each user. For example, the author of the topic has the ability to read, modify and delete the topic, while other users

only have the ability to read. If two topics are merged, this control information is lost. Instead merging two topics can be done in the following steps:

1. A topic checks the sub topics using the “Contains” association. If there are any two topics with the same URI, then begin to merge these two topics, otherwise there is no merging.
2. If the “Authored” association of the two topics connected to the same Person topic, two topics should be merged to a single topic. The result topic has the union of the characteristics of the two original topics including occurrences, associations, and names. The new topic will contain all the information that was in the original topics. The two original topics are deleted and then return to step 1 to check the new topic.
3. If the “Authored” association of the two topics connects to different Person topics, stop merging these two topics.

4.4 Privacy Implementation

Access control is important in our Context Management and Sharing system. In order for the end user to properly organize the content, they have to use existing resources which may be shared or belong to other users. Ideally every user has the privileges only necessary to do his job.

Management and sharing often relies on the different parts of the same XML document. Providing different access permissions on individual XML elements or sub trees will be expensive or impossible. We need to separate the management policy from the XML content and build up the architecture to efficiently manage the access control.

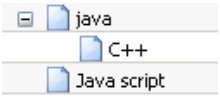
With the power of XML, enterprises can render content into any format and deliver it via multiple channels such as documents, Web pages, and wireless devices. XML can be used to describe the data and the hierarchical structure. This structure helps the policy writer to retrieve the portions of the XML document such as sub trees or sets of elements using the path expression. XPath is a popular expression language supported by the W3C [11].

In this paper, I use XACML to protect the content objects stored as XML documents. A simplified example of the content record is shown below:

```
<?xml version="1.0" encoding="UTF-8"?>
<presentation>
      <content id='"1"' creater='"001"' name='"java"'>
      <subtree>
            <content id='"3"' creater='"001"' name='"C++"'></content>
      </subtree>
      </content>
      <content id='"2"' creater='"002"' name='"Java Script"'></content>
</presentation>
```

The <presentation> element is used to display a tree structure of the content. It will be the root element when it is evaluated. It will contain one or multiple <content> element. The <content> element will contain zero or one <subtree> elements. The <subtree> element will contain more <content> element, and so on.

It will look like this:



The objective is to protect the content object. A content object may be owned by the current user or be shared by the other users. We want to assign different access rules for each content object. The precise rules are listed below:

1. The logged-in user can read and modify the attributes, modify structure, delete and add sub tree to the content object, but only if it is belong to the current user.
2. The logged-in user can read any public content object that is shared by the other users, but can not make any modifications.
3. The content object used for the system purpose can not be deleted, modified, added sub tree in any environment.
4. Any other access request is denied.

Rule 1 above can be translated into the following XACML code.

```
<?xml version="1.0" encoding="UTF-8"?>
<Rule RuleId="XACML-Rule1" Effect="Permit">
      <Description>A login user can modify his/her content
object</Description>
      <Target>
            <Subjects>
```

```
                  <Subject>
                        <SubjectMatch
MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
                              <AttributeValue
DataType="http://www.w3.org/2001/XMLSchema#string">login
user</AttributeValue>
                              <SubjectAttributeDesignator
AttributeId="urn:oasis:names:tc:xacml:1.0:user:user-group"
DataType="http://www.w3.org/2001/XMLSchema#string"></SubjectAttributeDes
ignator>
                        </SubjectMatch>
                  </Subject>
            </Subjects>
            <Resources>
                  <Resource>
                        <ResourceMatch
MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
                              <AttributeValue
DataType="http://www.w3.org/2001/XMLSchema#string">content</AttributeVal
ue>
                              <AttributeSelector
RequestContextPath="name(//content[1])"
DataType="http://www.w3.org/2001/XMLSchema#string"></AttributeSelector>
                        </ResourceMatch>
                  </Resource>
            </Resources>
            <Actions>
                  <Action>
                        <ActionMatch
MatchId="http://www.w3.org/2001/XMLSchema#string">
                              <AttributeValue
DataType="http://www.w3.org/2001/XMLSchema#string">read</AttributeValue>
                              <ActionAttributeDesignator
AttributeId="urn:oasis:names:tc:xacml:1.0:action:action-id"
DataType="http://www.w3.org/2001/XMLSchema#string"></ActionAttributeDesi
gnator>
                        </ActionMatch>
                  </Action>
                  <Action>
                        <ActionMatch
MatchId="http://www.w3.org/2001/XMLSchema#string">
                              <AttributeValue
DataType="http://www.w3.org/2001/XMLSchema#string">modify</AttributeValu
e>
                              <ActionAttributeDesignator
AttributeId="urn:oasis:names:tc:xacml:1.0:action:action-id"
DataType="http://www.w3.org/2001/XMLSchema#string"></ActionAttributeDesi
gnator>
                        </ActionMatch>
                  </Action>
            </Actions>
      </Target>
      <Condition>
            <Apply
FunctionId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
                  <Apply
FunctionId="urn:oasis:names:tc:xacml:1.0:function:string-one-and-only">
                        <SubjectAttributeDesignator
AttributeId="urn:oasis:names:tc:xacml:1.0:user:user-number"
DataType="http://www.w3.org/2001/XMLSchema#string"></SubjectAttributeDes
ignator>
                  </Apply>
                  <Apply
FunctionId="urn:oasis:names:tc:xacml:1.0:function:string-one-and-only">
```

```
                        <AttributeSelector
RequestContextPath="//presentation/content/"
DataType="http://www.w3.org/2001/XMLSchema#string"/>
                    </Apply>
                </Apply>
            </Condition>
</Rule>
```

In spite of these standard rules, the user can assign customized access rights to a content object. To set customized rights for the content object, select the content object in the tree, then right-click the “access setting” in the pop-up menu. In each row of the table, user can permit or reject the access. Figure 13 is an example:

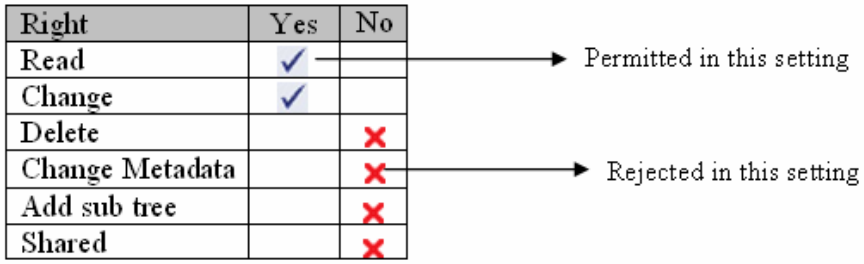


Figure 13: Customized Access Control

Because customized access rights may be different from the standard rules explained previously, these access rights will have higher priority over the standard rules. Access rights to a content object can be given to an individual user or a group that is built based on the same interests.

Access rights settings for a content object are automatically passed on to the sub tree content objects that the current object contains. That means that the access setting can be set once for the whole tree structure and the content objects will automatically inherit the access rights from their parent object.

5. Case Study

5.1 Overview

Thanks to the rapid growth of the Internet, it brings us tremendous amounts of Internet information. More and more users are focusing on how to make better use of

Internet information with a better user experience. In this chapter, we present a system we built for Management and Sharing of Context (MSC) based on Enriched Web Services Architecture. We want to prove that the Enriched Web Services Architecture is useful for our MSC system.

An Enriched Client should have the user interface with the look, feel and behaviour of modern desktop application. Bindows, a DHTML tool, provides the window controls that are missing in web applications, but are available in desktop applications. MSC should provide a rich user experience with the windows controls such as drag & drop, pop up menus, tree views, etc. MSC should be an integrated software application instead of jumping from one web page to another in the traditional web page. Web Services in DHTML will be a standard communication protocol to retrieve data from the server. As we have discussed in Section 3.1. The client will be divided into three layers: the Presentation tier, the Business Logic tier and Data Access tier. The Enriched Web Services Architecture of MSC should be more efficient than the Traditional Web Page Architecture in terms of response time, the number of request to the server and the amount of data transmitted. Also, it will have more features such as: cut and paste, drag and drop, tree view and tab browser, etc.

Topic Maps provide a more robust and semantically meaningful network as we have discussed in Section 4.3. XACML makes the user has the privileges only necessary to his job. It uses predefined policy to secure the web content. We want to see how Topic Maps and XACML can be used to manage the web context, secure it and share it.

Finally we will evaluate the Enriched Web Services Architecture of MSC against the Traditional Web Page Architecture of a test system. We want to see feature and performance improvement in MSC as a result of its Enriched Web Services Architecture.

5.2 User Interface

Choosing DHTML and JavaScript as the only acceptable UI (User Interface) for a context management system is becoming a major topic for discussion. The benefits include: they are easily distributed and upgraded, they are easily managed, and there is no need of installation. But it is a big problem that they no not solve the needs of

complex application UIs. Web user interfaces do not provide the quality of interaction required by professional users. While editing data inside a page and checking the data with the back end application, the whole page is reloaded. You will lose your focus.

Bindows has filled the UI gap with its lightweight Graphical User Interface Toolkit. “Bindows™ is a Graphical User Interface Toolkit for writing the rich client side web applications with the look, feel and behavior of modern desktop applications. Bindows applications are lightweight applications with zero footprint which means no installation required.”[1] It will provide a good platform for MSC. This web application need no installation, is easily upgraded and is accessible from everywhere.

A screenshot of the user interface can be seen in Figure 14.

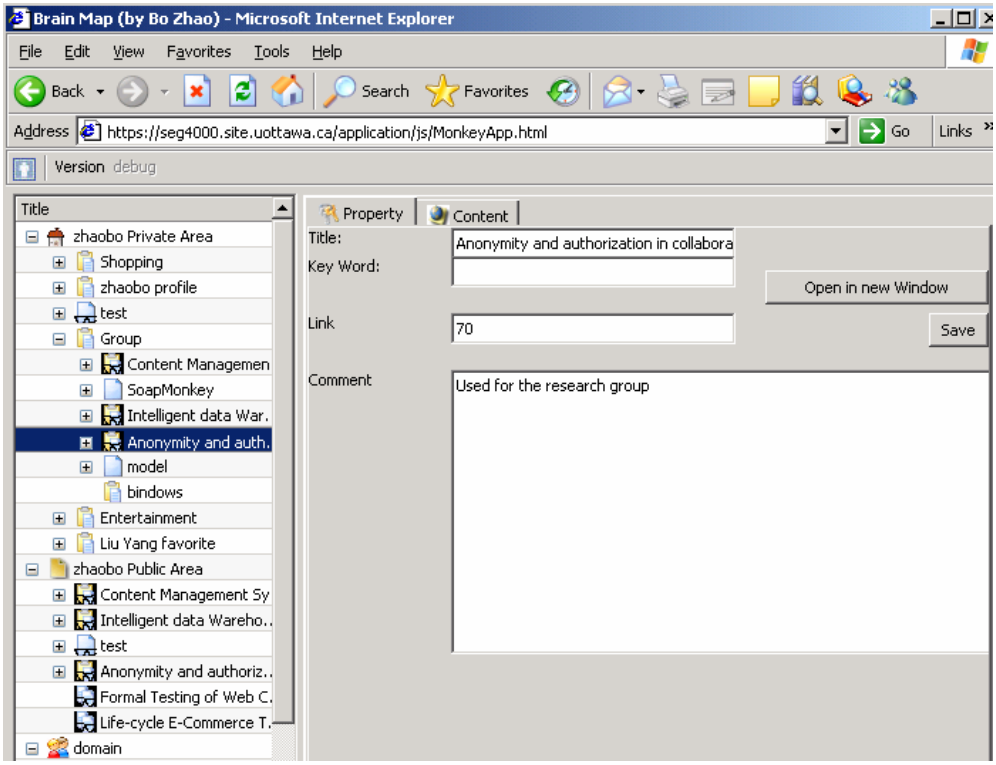


Figure 14 : the typical User Interface

The user interface consists of two windows: a navigating window and a content window. The left hand side is navigating window. It is organized as a tree view that makes the best use of user's screen and is a completely configurable display of expandable folders. Every node in the tree is a content object. The content object has specific attributes including: the name, resource, which standard template to be used to open the content object, etc. User can quickly navigate to any pre-saved content object and add, delete or rearrange the content object based on the user's organizational pattern. The right hand side is context window. It includes two tab windows: a property window and a Content window.

You can modify the attributes of each content object and open it. Figure 15 shows the content window.

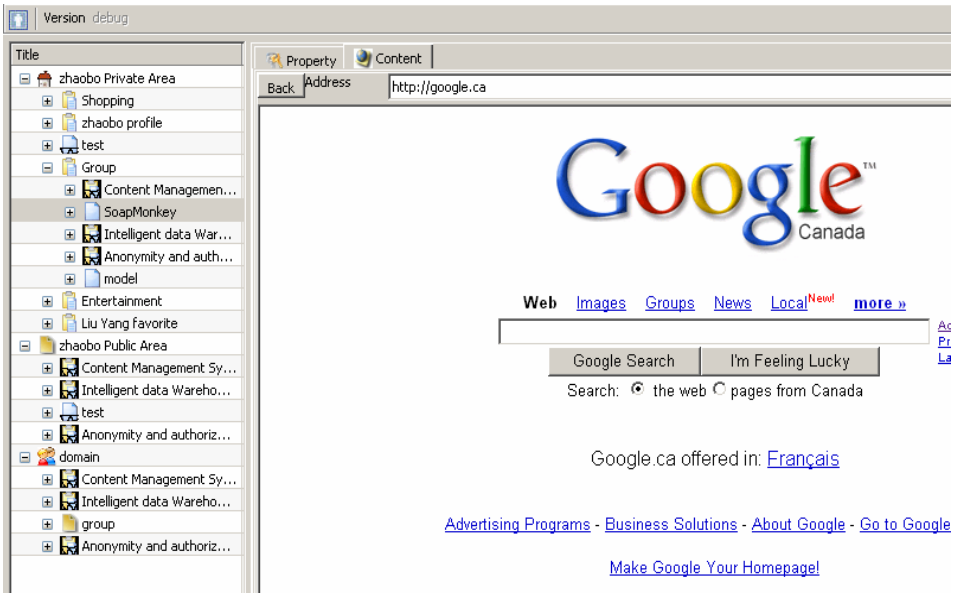


Figure 15 : the Content Window

5.2 System Architecture

This section will describe our system architecture for MSC (Figure 16). It includes three tiers: client side tier, business logic tier and data source tier.

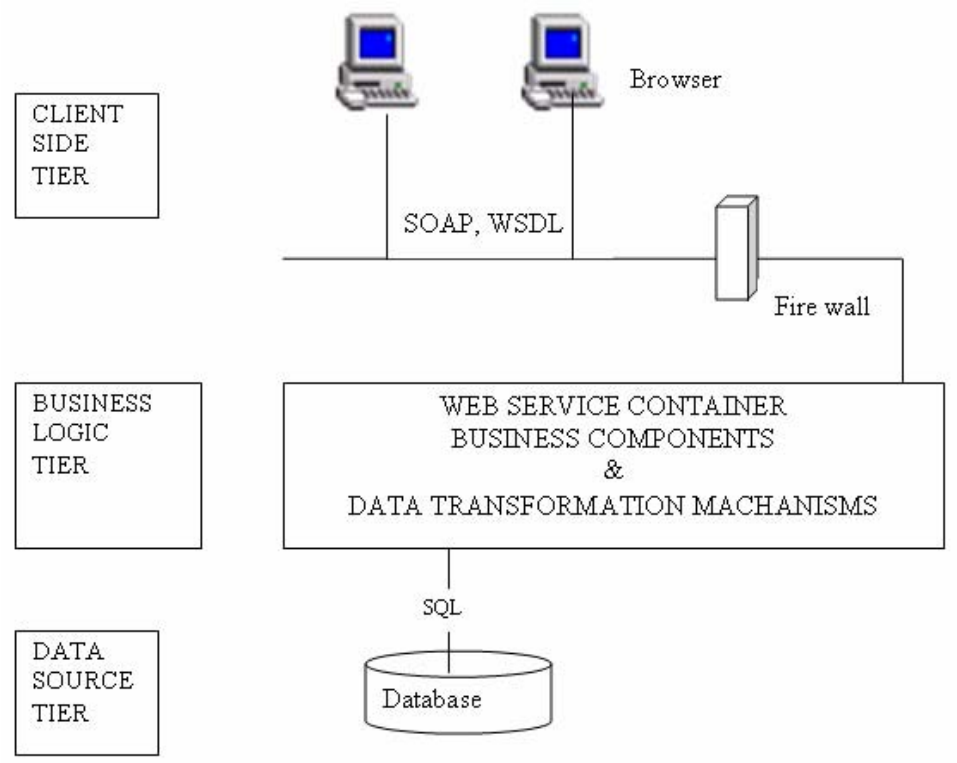


Figure 16: MSC System Architecture.

- The client side is Internet Explorer 6.0 using the Bindows platform. Because the Bindow platform is pure JavaScript and CSS, this platform can be automatically loaded and executed. IE can temporarily store the web resources in a cache then to be accessed later [29]. Mozilla and Netscape also provide browser caching capability. On each machine, the Bindows platform will be downloaded only once. That will reduce the Internet traffic and user's waiting time. While building up the user interface, the client creates a new instance of a Web Services proxy class which is ready to be used to call the Web Services.
- SOAP and WSDL. SOAP is designed to allow Internet communication between different programs. The two largest supporters of SOAP IBM and Microsoft both release their SOAP implementations. The client side, Jscript is from Microsoft. For the server side, we use IBM WebSphere Studio to

develop the business logic. Glue from WebMethods [31] can provide an interoperable Web Services between the client side and the server side.

- Business logic tier. We use IBM Websphere Studio 5.0 to develop some server side business logic. WebSphere5.0 only supports Java1.3. So some good packages in Java1.4 are not supported.
- Data Source tier. We use MySQL as the database. MySQL is an open source database. It is a very fast and robust multi-threaded relational database. MySQL is designed to work with small and mid-sized databases. The latest stable version of MySQL is 4.2.

5.4 Context Management

The navigating window provides functions to help users to manage (and share) their personal context. After you right-click the content object to display the popup menu, you can click on one of the items to choose the command. The following commands are supported by MSC.

- New: create a new content object from the existing template.
- Delete: delete the current content object.
- Share: make the current content object public to the other users.
- Copy/Cut: copy or cut the current node to the system clipboard.
- Paste: paste the node from the system clipboard to the current location.

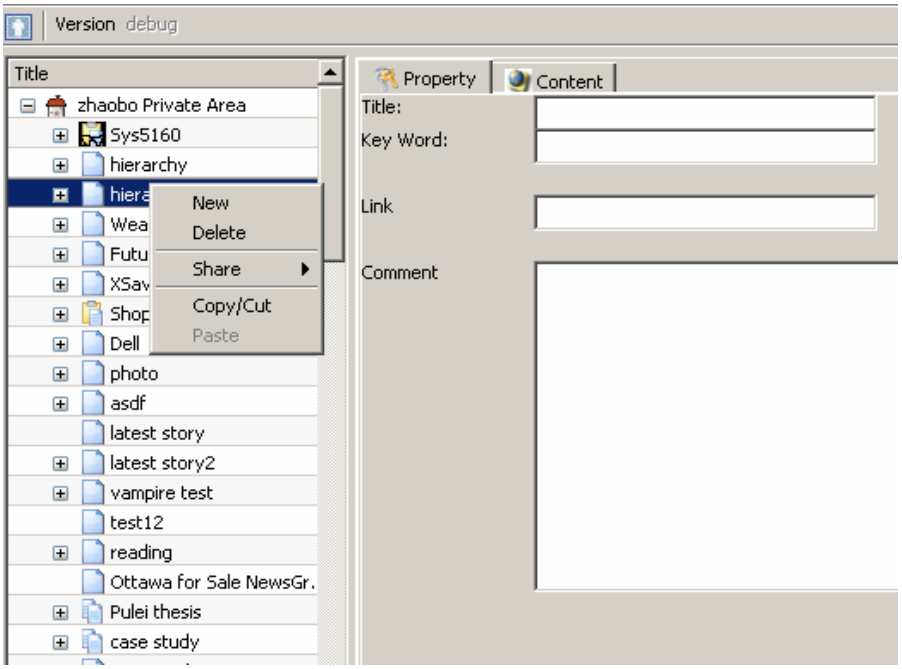


Figure 17 : Popup menu

When a user selects “New” command, a new window will be displayed. First a user must input a title for the content object, and then select one of three types of templates.

- Link: input a valid URL or paste one from the web page.
- File: upload a file to the server, the system will automatically create a content object with the link to the file.
- Note: a simply online edit tool appears.

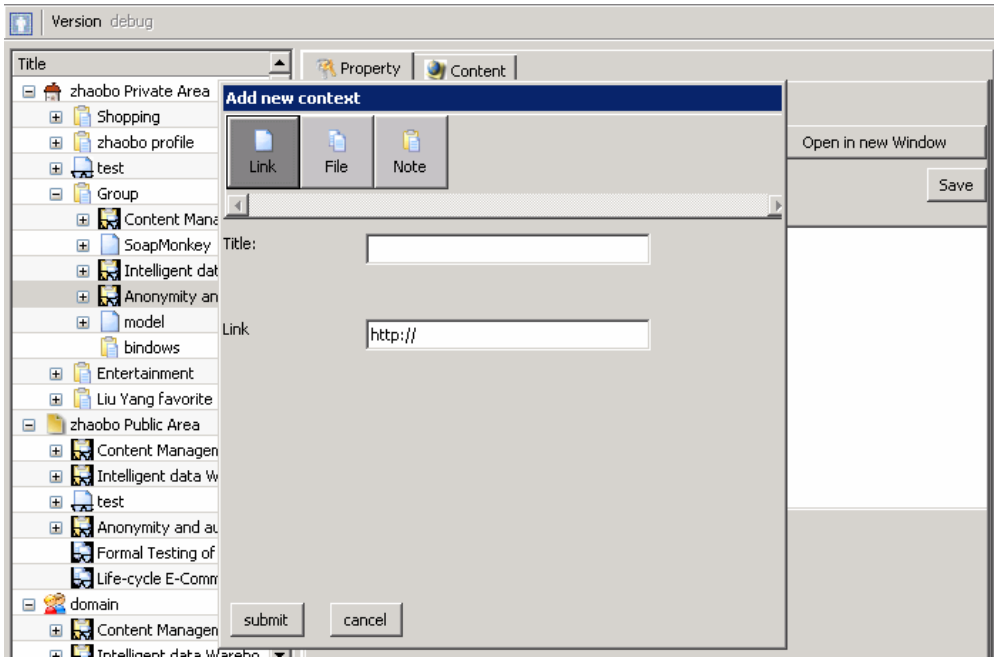


Figure 18 : New Command

After clicking the submit button, a new content object will be created under the current content object. The user can expand the current content to see the sub content.

From the left hand side tree view, the user can click on every content node to display the context of the current content on the right hand tab window. From the property tab window, a user can read or modify the content's context including title, key word, links and comment.

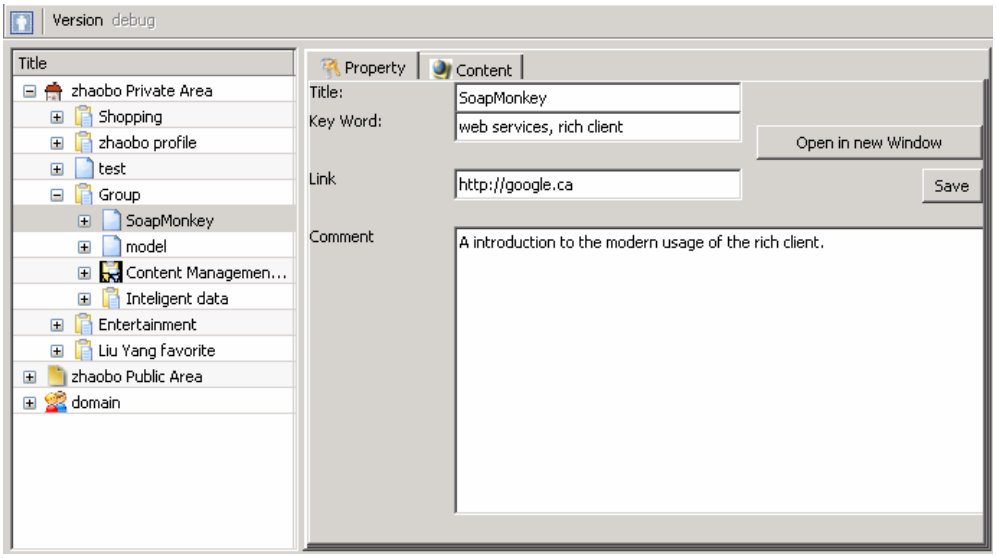


Figure 19 : Property Window

While the user clicks on the content tab window, the system will try to open the current content using a template according to the content's type. The link and file content will be opened using a browser.

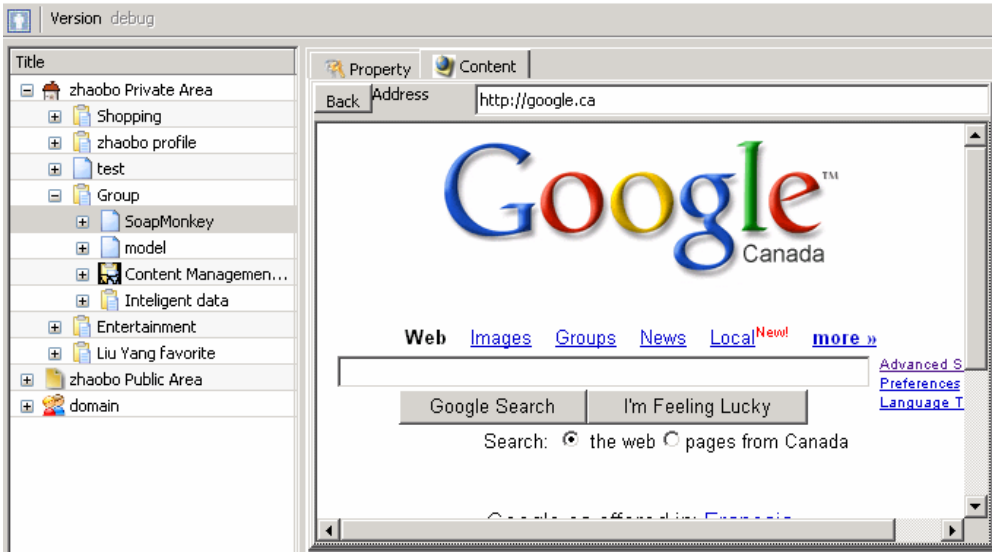


Figure 20 : Open a link content

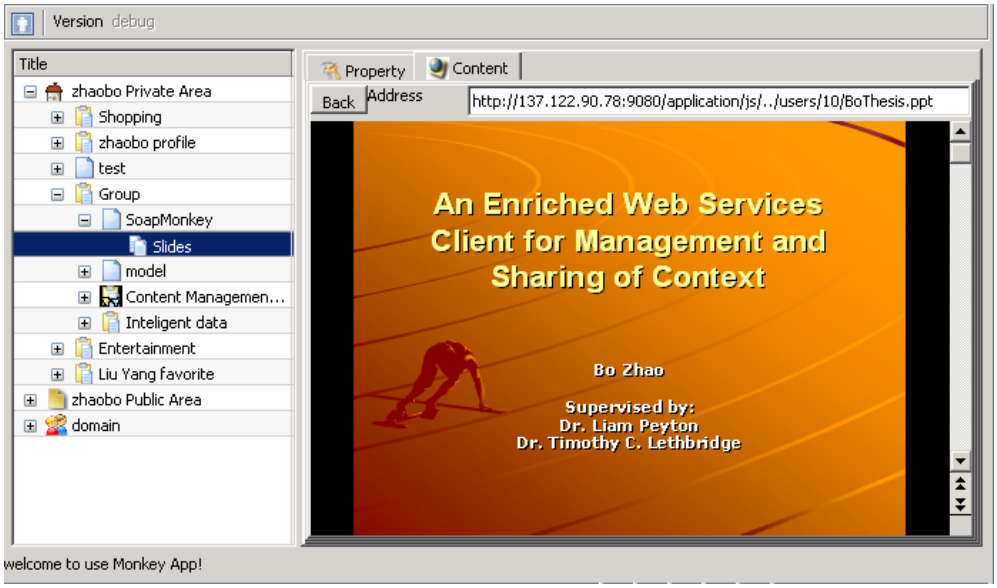


Figure 21 : Open a file content

When a user uploads a file to the server, the system will automatically create a file content object with the link to the file. It will be opened in a browser or downloaded to the local file system. This feature will help the user to organize their personal document or pictures.

Figure 22 is the screen shot when a user opens note content. MSC opens the note content using an HTML editor. This editor provides some basic functions such as font formatting (bold, italic and underline), copy and paste.

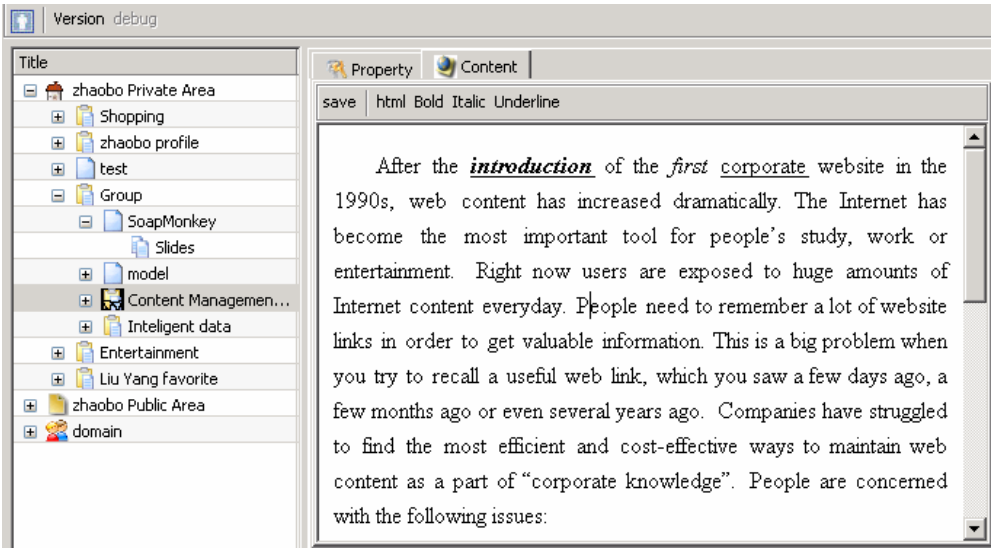


Figure 22 : Open a note content

MSC provides drag and drop function. Figure 23 shows the screen shot of drag and drop. Also, the user can right click the content and select copy/paste items from the popup menu.

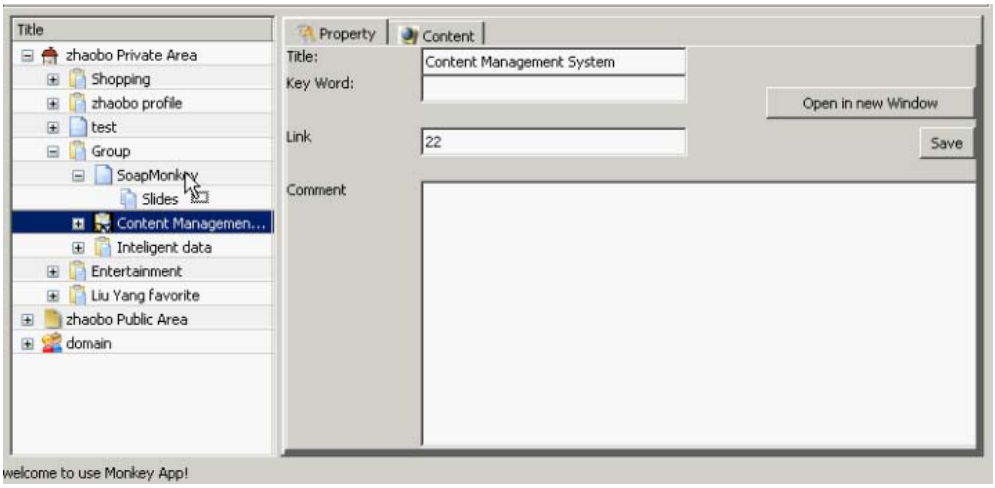


Figure 23 : Drag and Drop

5.5 Context Sharing

If the user right clicks the content object, a popup menu will be displayed. A user can select Share->public item to make the current content to be public to all the other users.

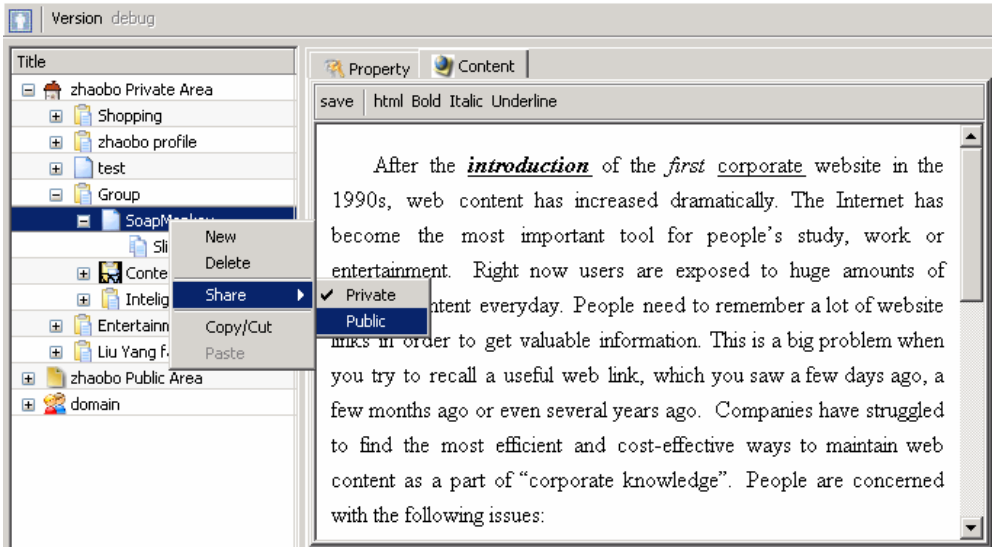


Figure 24 : Set the content to be public

After sharing the content, other user can find out all the shared content under the domain node.

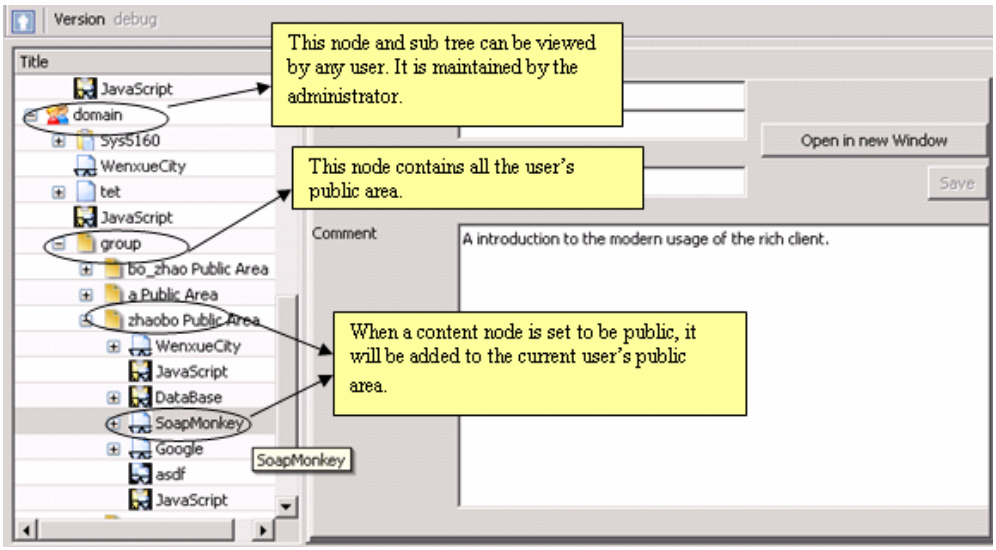


Figure 25 : Share the content node

From Figure 26, we can see that the user can view other's shared content node and add them to their private area using copy and paste functions. The user right clicks on the content node, and then selects the Copy/Cut item from the popup menu.

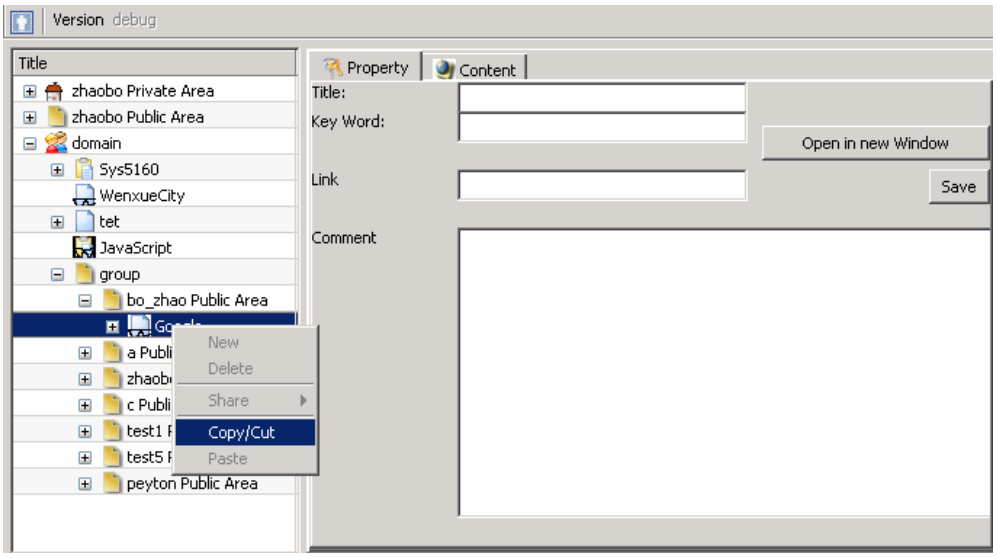


Figure 26 : Copy the content node

In the user's private area, the user can right click on one content node and select Paste item from the popup menu.

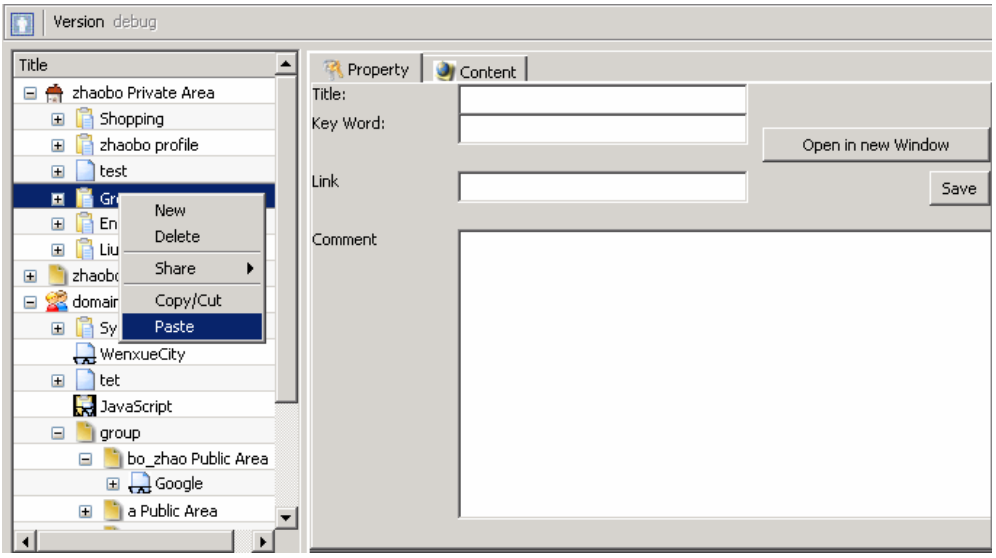


Figure 27 : Paste the content node

The Paste Type dialog will appear. The user can create a link of the content node, copy a single content node, or copy the whole sub tree of the content node. In this way, the user can collect all interested content in one place.

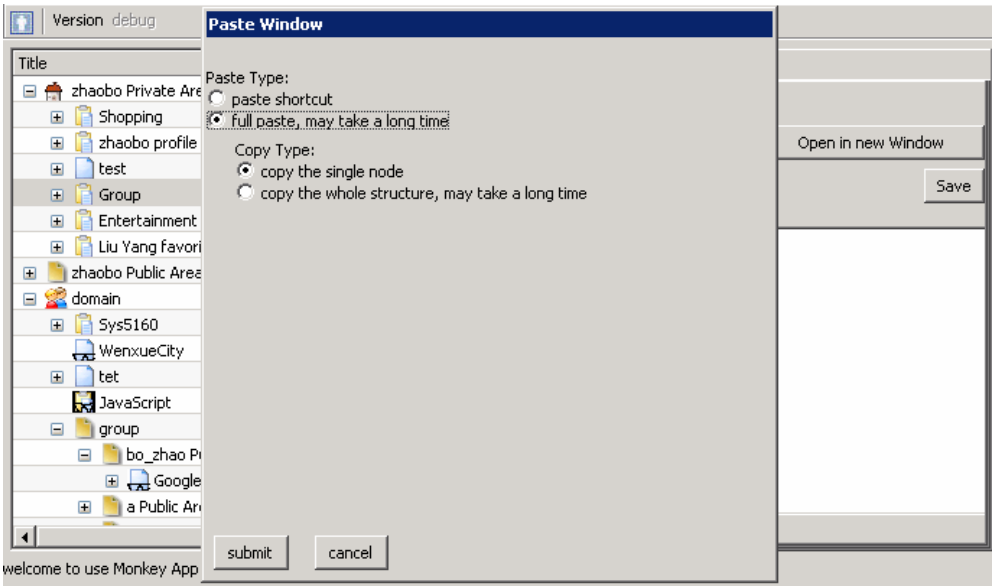


Figure 28 : Paste type dialog

5.6 Key Mechanisms for Dynamic Updating of the UI

An Enriched Web Services Client gives a rich user experience in a browser application. The key feature of an Enriched Web Services Client is the ability to update the user interface dynamically. In this section, we will describe key mechanisms for dynamic updating of the user interface which distinguish MSC from a Traditional Web Page system.

5.6.1 Structural Cut and Paste

In Bindows, the pop up menu is supported. When the user right clicks the tree node, a mouse event is fired to display a pop up menu. The user can select the items from the pop up menu which are interpreted as operations on the underlying XML Data Model.

When the user selects the Cut action from the pop up menu, the current tree node is saved into the system's clipboard. When the user right clicks another tree node, a pop up

menu is displayed according to the current system settings. For example, if the system's clipboard is empty, the Paste action is disabled. Otherwise the Paste action is enabled.

If the user selects the Paste action in the pop up menu, MSC does the following steps:

1. Removes the source tree node from the underlying XML Data Model.
2. Adds the source tree node to the target tree node as a child in the underlying XML Data Model.
3. Sends an asynchronous request which will update the data in the server.
4. Updates the tree view according to the changed XML Data Model.

5.6.2 Drag and Drop

Drag and Drop is like Cut and Paste but it has to be completed immediately. In Bindows, Drag and Drop is event based. The mouse events are interpreted as actions on the underlying XML Data Model and the tree view is updated according to the changed XML Data Model.

When the user holds the mouse on a tree node and then moves the mouse a small distance without releasing the mouse button, a dragstart event is fired. This event saves the current tree node, changes the mouse icon to reflect that the Drag and Drop process has started.

When the mouse pointer enters a potential drop target, a dragmove event is fired. This event checks whether the drop target is a tree node, then reads the allowDrop property of the tree node. If the allowDrop property is false, MSC changes the mouse icon to reflect that it is not a possible drop target.

A dragdrop event is fired when the user release the mouse button. This event does the following steps:

1. Stop the Drag and Drop process if the source tree node and the target tree node are the same. Stop the process if the target tree node is the parent of the source tree node.
2. Remove the source tree node from the underlying XML Data Model.
3. Add the source tree node to the target tree node as a child in the underlying XML Data Model.

4. Send the asynchronous request which will update the data in the server.
5. Update the tree view according to the changed XML Data Model.

5.6.3 Asynchronous Server Interaction

There are two types of method call: asynchronous and synchronous. An asynchronous method call will return immediately so that the user can perform other operations while the called method completes its work later through a callback function. A synchronous method call will block the user interface until the called method is completed.

The client side contains the XML Data Model. When a request comes from the Presentation component, MSC first looks in the XML Data Model. If the requested data can be found in this step, MSC will take care of updating the XML Data Model when needed, returning the result immediately and sending an asynchronous method call to update the data in the server. In the Presentation tier, an asynchronous method call is more desirable, because it will not block the user interface. In MSC, the asynchronous method calls include: add, delete, modify etc.

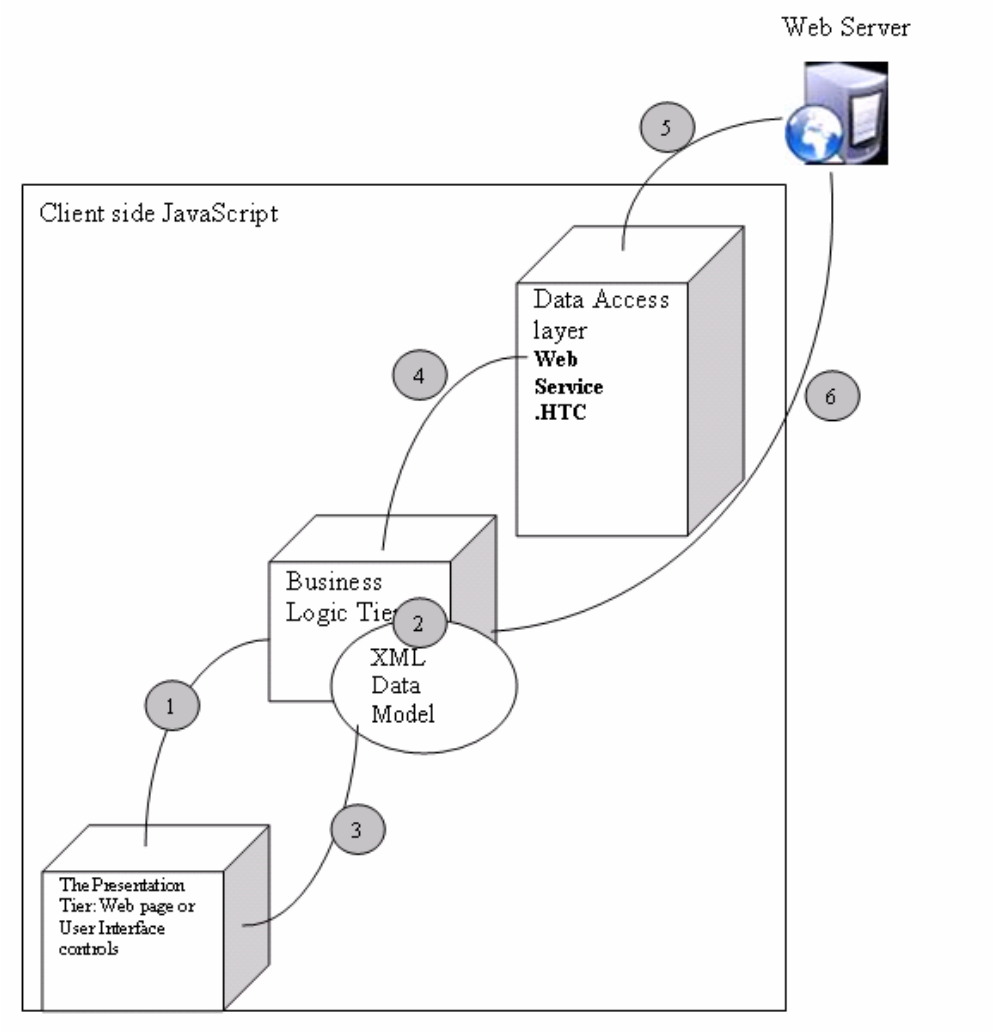


Figure 29 : Asynchronous Server Interaction

1. According to the user's operations on the user interface, the system sends the request to the Business Logic Tier.
2. The Business Logic Tier will update the XML Data Model when necessary.
3. The request will be returned immediately and the Presentation Tier will update the user interface.
4. If the XML Data Model has been changed, the Business Logic Tier sends a request to the Data Access Tier to update the data in the server.

5. The Data Access Tier sends an asynchronous method call to the server through the Web Services.
6. The server sends back the result. Through the callback mechanism, the Business Logic Tier will confirm the update to the XML Data Model which is done in step 2.

6. Analysis of the Results

6.1 Overview

In this chapter, we describe an experiment to compare the performance of the Enriched Web Services Architecture implementation of MSC with a Traditional Web Page Architecture implementation. The results of this experiment provide some insight into the advantages of Enriched Web Services Architecture for Management and Sharing of Context.

"In a traditional web page, a user would typically select a link on the page presented in the browser, which would lead to another HTTP request."[47] The client browser navigates to a new URL, renders the HTTP response and displays the entire page. The web application navigation is designed as a set of URL links that invokes either a JSP or a static HTML page on the server. The client side will not include Business Logic Layer, web services. There are no Anchors, images etc in the page. Also any other technologies that are not zero footprints will not included such as DOM, Java Applets, Flash and ActiveX.

In order to compare the Enriched Web Services Architecture of MSC with a Traditional Web Page Architecture, we built a "test" system using a Traditional Web Page Architecture. The user selects a link on the page, which leads to another HTTP request returning a completely new web page. The HTTP request normally invokes either a JSP or a static web page on the server. Figure 30 is the screen shot for the "test" system.

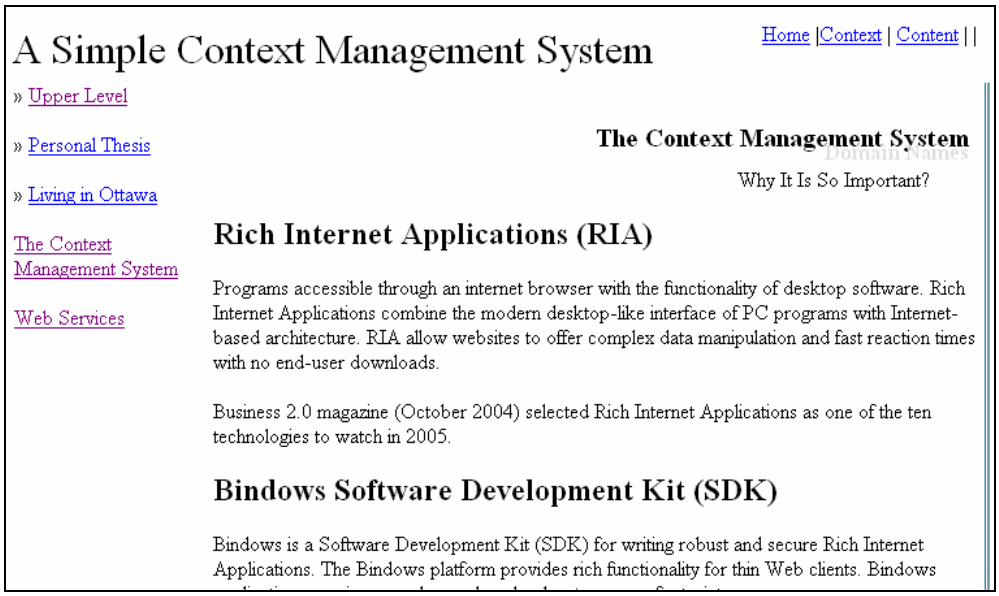


Figure 30 : a simple context management system

The purpose of this experiment was to compare the performance of the Enriched Web Services Architecture used by MSC with the Traditional Web Page Architecture of the "test" system. We wanted to measure the response time, the number of requests to the server, the amount of data transmitted. Also, we wanted to make a feature comparison. In Chapter 5, we have used some advanced controls or functions such as: tree view, drag & drop, popup menu. We wanted to see whether these advanced controls or functions are possible in the "test" system.

6.2 Test Scenarios

The following test scenario is to simulate a single user's interactions with the Enriched Web Services Architecture of MSC and the Traditional Web Page Architecture of the test system. The scenario will be incorporated into a test script to test the system's performance. There are ten typical interactions. The test scenario simulates a user's entire set of interactions in the system with different frequencies.

Interaction steps	Frequency	Description
1. Log in	1	A user logs in and begins to use the system
2. View the context of the content	3	Read the detailed context and attributes of the current content. These information includes the creator, the topics it contains, the URL, the control access
3. Modify the context of the content	2	Modify the detailed context and attributes of the current content.
4. Read the content	3	If it contains URL, use web browser to open it. Otherwise use HTML editor to open it.
5. Delete the content	2	Delete the current content and all the sub content.
6. View the sub content	3	In the content tree, expand the current content to see the sub content.
7. Create a new content	2	Begin to create a new content.
8. Select the content type	2	Select a pre-defined template to create a new content.
9. Add the context of the content	2	Input the context and attributes to the current content.
10. Set the current content to be public	2	Put the content into the public area, and all of the users can read the content.
11. Log out	1	A user log out and leave the system.

Table 1 : user's entire set of activities in the system

One test scenario will be one experiment. After 30 experiments, three key criteria are collected and these data is used to analyze the performance of the system.

The following explains the key definitions used when analyzing the website.

1. Request count. In the Traditional Web Page Architecture, every user's click on the web page will invoke a HTTP request to the server. In the Enriched Web Services Architecture, not every click will invoke a request to the server. The Enriched Web Services Architecture will reduce the network traffic and server's work load.
2. Response time. In the Traditional Web Page Architecture, the user should wait for the HTTP response before making their next move. In the Enriched Web Services Architecture, if the user can get data from the XML data model,

there is no waiting time. Otherwise the user should wait for the Web Services response.

3. The amount of received data. The two architectures are compared to see how much internet traffic will be used.

After 30 experiments, the average data for each step of the test scenario is collected for every key criterion. The analysis results are showed in the following sections.

6.2.1 Features and Functionality Comparison

The following are the key comparison points between the Enriched Web Services Architecture of MSC and the Traditional Web Page Architecture of the test system. The last three comparison points are based on the scenario of Section 6.2. A Traditional Web Page may contain JavaScript, HTML or JSPs. Through the HTTP request process, the current web page is destroyed and a new web page is redrawn completely on every user's action if we do not talk about the browser caching. The Enriched Web Services Client of MSC is the DHTML with the Web Services support. It retrieves data from the server through Web Services and updates the user interface partially. It significantly decreases the response time and the Internet traffic.

	The Enriched Web Services Architecture of MSC	The Traditional Web Page Architecture of test system
1. Connect to different services	Yes	No
2. Cut and Paste	Yes	Limited
3. Drag and Drop	Yes	No
4. Tree View and Tab Browser	Yes	Limited
5. Pop up menu	Yes	Limited
6. The total requests count to the server	13	23
7. The total response time	38s	210s
8. The total amount of data transmitted	48.1 kb	198.4 kb

Table 2 : Feature Comparison

The following list explains the Table 2. The list briefly explains the meaning of the compared feature and how we get the comparison results.

1. Connect to different services. In the Traditional Web Page Architecture, the web content is returned from only one server. In the Enriched Web Services Architecture of MSC, JavaScript can be used to call the Web Services from different servers and combines the results to update the user interface.
2. Cut and Paste. The Traditional Web Page Architecture supports Cut and Paste of text. In the Enriched Web Services Architecture of MSC, we support structural Cut and Paste of objects.
3. Drag and Drop. The Traditional Web Page Architecture does not support this feature.
4. Tree View and Tab Browser. The Traditional Web Page Architecture can contain a tree view or a tab browser. But they are redrawn completely on every user's action. In the Enriched Web Services Architecture of MSC, the tree view and the tab browser can be reused and dynamically updated.
5. Pop up menu. In the Traditional Web Page Architecture, the pop up menu is supported by JavaScript. But the look or feel of it can not be controlled. It is just a message with acknowledgement, not a full dialog. In the Enriched Web Services Architecture, a pop up menu are linked to the underlying XML Data Model and interpreted as actions on the XML Data Model.

6, 7 and 8 will be explained in the following sections.

From Table 2, we can see that the Enriched Web Services Architecture provides more advanced features and functions that are not available in the Traditional Web Page Architecture.

6.2.2 Total Request Count to the Server

In the Traditional Web Page Architecture, every user's click on the web page will invoke an HTTP request to the server. In the Enriched Web Services Architecture of MSC, the client holds an XML Data Model object. When the user make a mouse click on the UI (User Interface), the client will look for the data in the XML Data Model. If not

found, the client will make a SOAP request to the server. Figure 31 shows the request count for each step in the test scenario. The X axis is the interaction step number from the test scenario. The Y axis is the number of requests to the server. From Figure 31, we can figure out that there are fewer requests in the Enriched Web Services Architecture of MSC than the Traditional Web Page Architecture. Most of the user's clicks will get the result on the client side in MSC. This feature will reduce the number of the Internet requests and the server's workload.

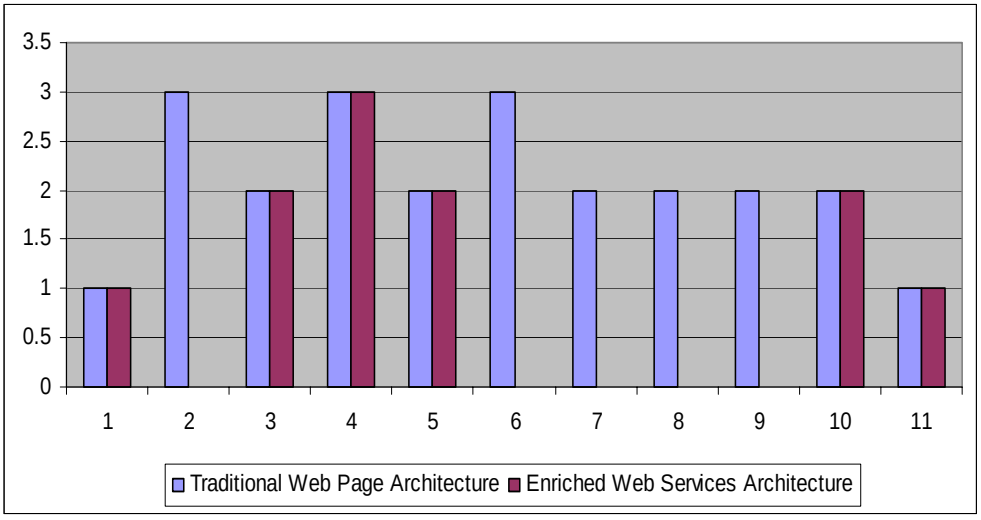


Figure 31 : Total request count to the server for each interaction step (Y axis)

Figure 32 shows the total requests count for both of these two architectures. In the Traditional Web Page Architecture, one interaction step will lead to a HTTP request to the server. In the Enriched Web Services Architecture of MSC, some interactions can be completed in the client side.

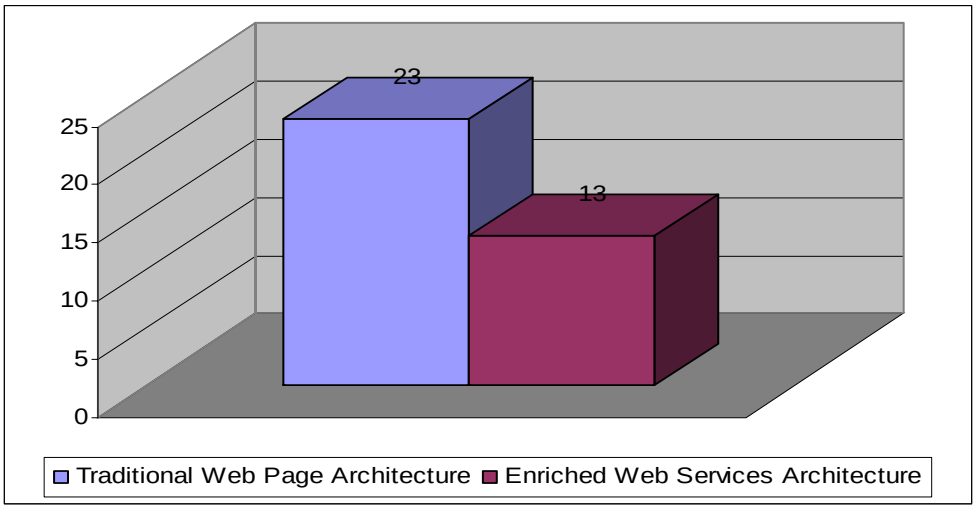


Figure 32 : Total Request Count to the Server

6.2.3 Total Response Time

Every HTTP request will need several seconds to get the response. The actual response time depends on the website. In the traditional website, the user will wait for several seconds before making their next move. In MSC, some data can be retrieved from the local XML Data Model and the UI (User Interface) can get the data immediately. Also, some interactions will send asynchronous Web Services calls and the user interface can be updated immediately. For the user, this means zero wait time for the click response. Figure 33 is the response time. The X axis is the interaction step number from the test scenario. The Y axis is the total response time and measure unit is million seconds.

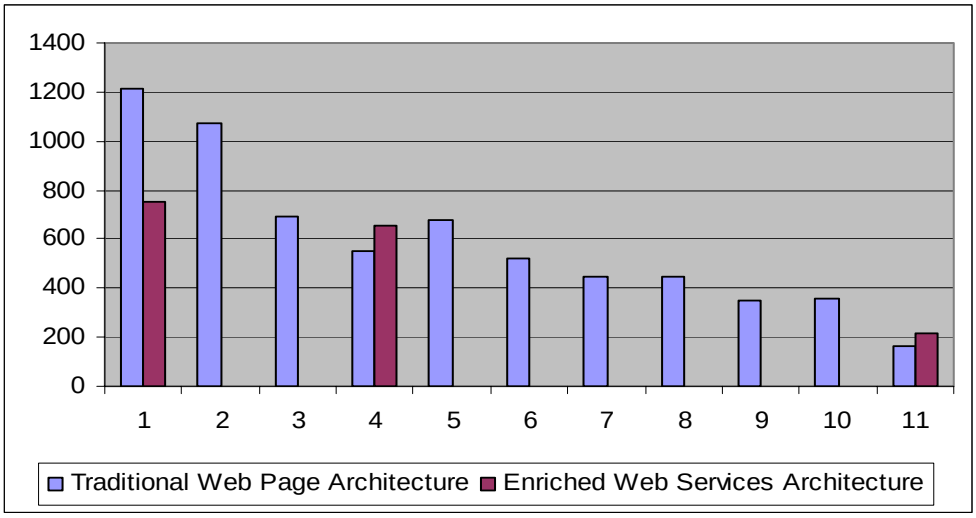


Figure 33 : Total Response Time in Million seconds (Y axis), for each interaction step (X axis)

From Figure 34, we can clearly see that the user has lower response time. This feature makes the user's interaction efficient and quick.

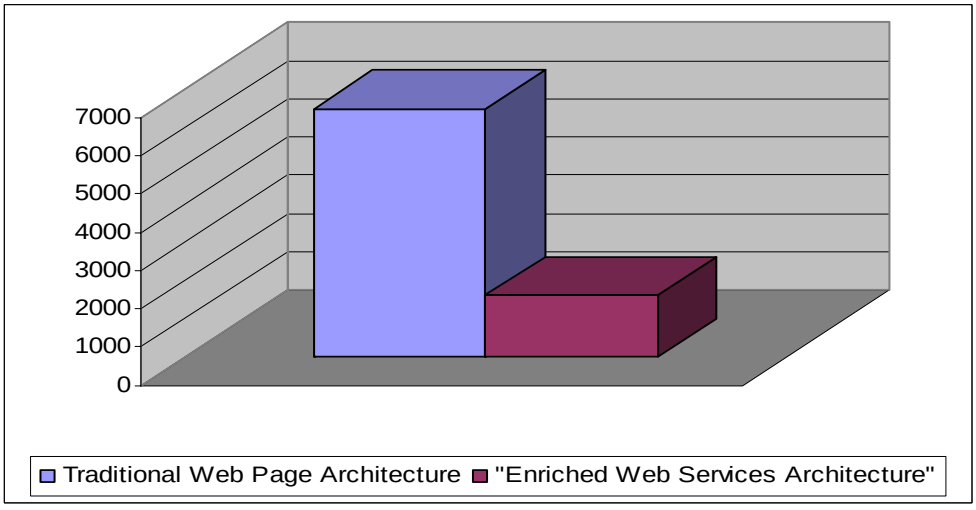


Figure 34 : The Total of the Response Time

6.2.4 Total Amount of Received Data

Figure 35 shows the amount of received data from the server. The Enriched Web Services Architecture of MSC will hold a document object. All the following SOAP requests will continually add the new data into this document object. If the client can find the data in this document object, it will not send the request to the server. As time passes by, the client will hold most of the data that the user needs. The total data traffic will be reduced.

Figure 35 shows on the Y axis the amount of received data from the server in Kilobytes. The X axis is the interaction step number from the test scenario.

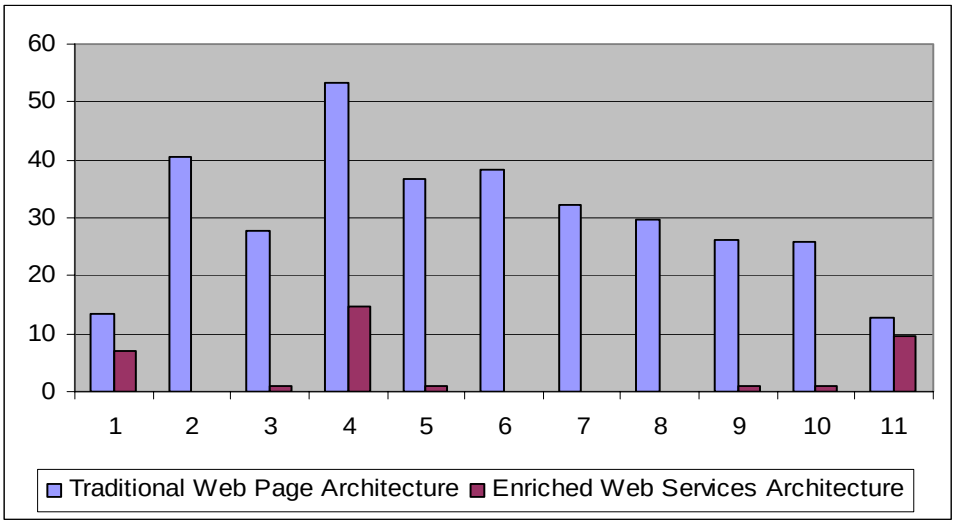


Figure 35 : Total Amount of Received Data in KB (Y axis), for each interaction step (X axis)

From Figure 36, we can clearly see that the Internet traffic is reduced.

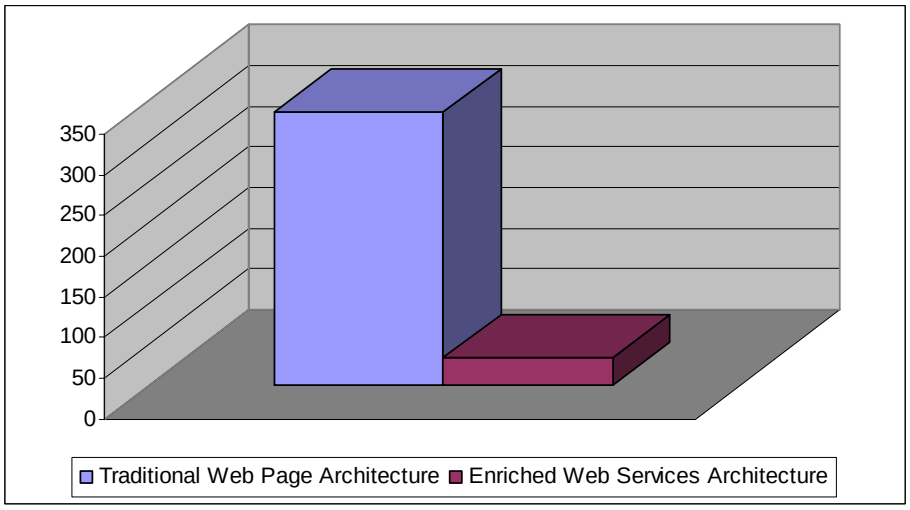


Figure 36 : The Total of the Received Data

7. Conclusion and Future Work

7.1 Summary

Today's rapid growth of the Internet content has given users too many information resources as well as too much unstructured data. Users need new tools to better organize the available information found in worldwide resources, and with a better user experience.

This thesis presented an architecture and example system (MSC) illustrating the integration of DHTML with Web Services for the management and sharing of context. The main contribution of this thesis is to build what we call the Enriched Web Services Architecture. This architecture will help to reduce the internet traffic and increase the system performance.

DHTML is used give the rich user experience in a zero foot print web browser. Web Services provides a standard XML based communication protocol. Topic Maps and XACML are used to manage the context. Finally, we compare the Enriched Web Services Architecture of MSC with the Traditional Web Page Architecture to see the performance and features improvement.

7.2 Conclusions

As we have discussed in Section 3.1, the client is divided into three layers: the Presentation tier, the Business Logic tier and Data Access tier. Each tier can be developed and deployed independently. The client maintains the XML Data Model and sends the requests to the server when more data needs to be aggregated into the XML Data Model. Also, the client separates the visual presentation from the content. The user interface can be dynamically updated and the same data can be displayed in various formats.

In conclusion, this architecture shows much improvement over the Traditional Web Based architecture. From the analysis of results in Chapter 6, we can see that an Enriched Web Services Architecture improves system performance. The number of requests to the server almost drops by half. The waiting time in the user interface is significantly decreased with many operations appearing to be instantaneous to the user. The amount of data transmitted is decreased. Most importantly, the Enriched Web Services Architecture provides more features such as: drag and drop, popup menus, cut and paste, etc. It enables the creation of more complex interfaces for both developer and user and provides more capability to fulfill the current e-commerce needs.

Topic Maps are used to build a semantically meaningful network among Internet documents. It provides a robust and context-aware architecture for management and sharing context. The context of a topic includes most of the information in which a user is interested such as: the creator, the topics it contains, the web link, the control access, etc. This information can be easily combined into the Topic Maps and build a semantic network. Some Topic Maps manipulation methods are discussed in Chapter 4 and implemented in the case study. It provides a good architecture about how to manage the context using Topic Maps.

In order to protect the privacy and enforce the security requirements, XACML is studied in Section 4.4. It makes the user has the privileges only necessary to his job.

7.3 Issues and Future Work

However, some technical problems still lie ahead. In our system, we already define some topic types and association types in Topic Maps. But in order to describe the real

world objects, they are not enough. The better approach is to build a topic type hierarchy like the class hierarchy in Java language. In this way, we can better organize our knowledge in a semantically meaningful network.

For the future, MSC should support more templates that will support most people's daily activities, such as: calendar, personal schedule list, build-in content context search. Another challenge is to distribute the Content Management System to the different machine and still provide cross references.

8. References

[1] Bindows, URL: http://www.bindows.net/

[2] Terveen, L.G., and Hill, W.C. *Beyond Recommender Systems: Helping People help Each Other*, in Carroll, J. (ed.), HCI in the New Millennium, Addison-Wesley, 2001.

[3] Sharon T., Lieberman H., and Selker T., *Searching the Web with a Little Help From your Friends*, Poster to be published in CSCW, 2002.

[4] Rath, H H. *XML Topic Maps for Knowledge Management*. URL: http://www.empolis.com/downloads/empolis_TopicMaps_Whitepaper20030206.pdf Sydney, SAI Global, 2004.

[5] Brin, S. and Page, L. *The Anatomy of a Large-Scale Hypertextual Web Search Engine.* Computer Networks and ISDN Systems, 30(1-7), 1007-117. 1998

[6] Page, L. Brin, S. Motwani, R. and Terry, W. *The PageRank Citation Ranking: Bring Order to the web.* Technical Report, Stanford University, 1998.

[7] Balabanovic, M. and Shoham, Y. *Content-Based, Collaborative Recommendation.* Communications of the ACM, Vol. 4, No. 3, 1997.

[8] Paul, S. *It's the Context, Stupid*. Wired Magazine Issue 2.03, Mar 1994.

[9] Parson, J. *Building a high-performance content infrastructure with Web Services*, Computer Technology Review, 2003. Los Angeles. June 2003.

[10] Peyton, L. *Measuring and Managing the Effectiveness of Personalization*, Proceedings of the 5th international conference on Electronic commerce, p.220-224, Pittsburgh, Pennsylvania. Oct 30, 2003.

[11] W3C XML Path Language(XPath) 2.0. URL: http://www.w3.org/TR/xpath20/ Feb, 2005.

[12] Masahiro, H. and Hideaki, T. *Neighborhood Matchmaker Method: A Decentralized Optimization Algorithm for Personal Human Network*, Lecture Notes in Computer Science, Volume 2773, p. 929 – 935, Oct 2003.

[13] Jacob, N. *Designing web usability*. New Riders Publishing, 2000.

[14] W3C DOM, URL: http://www.w3.org/DOM/, 2005.

[15] Henrik, A. Johan, G. *Collaborative development of ontologies in a Peer-to-Peer environment*. URL: http://www.kvocentral.com/reports/henrikandjohan.pdf. Brisbane, Oct 2002.

[16] Ramana, V. *The importance of hierarchy building in managing unstructured Data.* URL: http://www.kmworld.com/publications/whitepapers/ECM02/venkata.pdf . March, 2002.

[17] Ian, L. *A Web Content Management Blueprint: planning for a content-rich, successful web site.* URL: http://www.portentinteractive.com/library/cmsexplained.pdf. Portent Interactive, 2002.

[18] Dimitrios, M. *Open Source Content Management Systems: An Argumentative Approach.* URL: http://michelinakis.gr/Dimitris/cms/oscms-report.pdf. August, 2004.

[19] King, S, *Web content management*, Computer Technology Review, Los Angeles. Vol. 22, Iss 11, 9. Nov, 2002.

[20] Parson, J. *Building a high-performance content infrastructure with Web Services*, Computer Technology Review, Los Angeles. Vol. 23, Iss 6, 34, Jun 2003.

[21] *About the WebService Behavior*. Microsoft Developer Network, URL : http://msdn.microsoft.com/library/default.asp?url=/workshop/author/webservice/overview.asp

[22] Kreger, H. *Web Services Conceptual Architecture(WSCA 1.0).* URL: http://www-4.ibm.com/software/solutions/webservices/. May, 2001.

[23] Gottschalk, K. et al., *Web Services architecture overview: the next stage of evolution for e-business.* URL: http://www-106.ibm.com/developerworks/library/w-ovr. September,2000.

[24] Johnson, T. Mathews T, George G. *Modeling of Web Services Flow*. IEEE International Conference on E-Commerce(CEC'03). USA. 2003.

[25] Jan Rihak. *Access Control Markup Languages For XML Documents.* Swiss Federal Institute of Technology Zurich. August 2004.

[26] Jerome M. *Personal Knowledge Management: The Basis of Corporate and Institutional Knowledge Management.* URL: http://www.spottedcowpress.ca/KnowledgeManagement/pdfs/06MartinJ.pdf. 2000.

[27] Jason, F. Carol, H. *Personal Knowledge Management : Who, What, Why, When, Where, How?* URL: http://www.anderson.ucla.edu/faculty/jason.frand/. Dec, 1999.

[28] Sun. *Sun's XACML Implementation.* URL: http://sunxacml.sourceforge.net/. 2003.

[29] Michael, K. and Prashant J. *Caching*. URL: http://www.cs.wustl.edu/mk1/Caching.pdf. Germany, 2002.

[30] Jame Snell. *The Web Services inside, Part 3: Apache and Microsoft—playing nice together. SOAP interoperability is coming around.* URL: http://www-106.ibm.com/developerworks/library/ws-ref3/?n-ws-5241. May, 2001.

[31] WebMethods. URL: http://www.webmethods.com/

[32] Arasu, A. et.al. *Searching the Web.* ACM Trans. On Internet Technology, Vol. 1, No. 1, p. 2-43. Aug, 2001.

[33] S. Hada and M. Kudo. *XML Access Control Language: Provisional Authorization for XML Documents*, Tokyo Research Laboratory. IBM Research. Oct 16, 2000.

[34] OASIS. URL: http://www.oasis-open.org/.

[35] Thijs van den Berg, Marya Steenman. *XACML/WSPL*. URL: http://www.os3.nl/~mrtn/assignments/XACML.pdf. Dec, 2004.

[36] OASIS. *eXtensible Access Control Markup Language(XACML) Version1.0.* URL: http://www.oasis-open.org/committees/download.php/2406/oasis-xacml-1.0.pdf . Feb, 2003.

[37] OASIS. *eXtensible Access Control Markup Language(XACML) Version2.0.* http://docs.oasis-open.org/xacml/access_control-xacml-2_0-core-spec-cd-04.pdf Dec, 2004.

[38] Steve P and Graham M, editors. *XML Topic Maps (XTM) 1.0.* TopicMaps.Org, 2001. URL: http://www.topicmaps.org/xtm/index.html#goals

[39] *XML Topic Maps (XTM) Processing Model 1.0.* TopicMaps.Org, 2000. URL: http://www.topicmaps.org/xtm/1.0/xtmp1.html

[40] *The Curl Client/Web Platform.* 2001, Curl Corporation. URL: http://www.curl.com/pdf/The_Curl_ClientWeb_Platform.pdf

[41] *About the WebService Behavior.* 2005, Microsoft. URL: http://msdn.microsoft.com/workshop/author/webservice/overview.asp

[42] M.Lorch, S.Proctor, R. Lepro, Kafura D, and S.Shah. *First Experiences Using XACML for Access Control in Distributed System.* October 2003.

[43] W3CSchools, *Introduction to DHTML*. URL: http://www.w3schools.com/dhtml/dhtml_intro.asp

[44] M. Stock. Technologies for Thin Client Architectures. Jan 2001. URL: http://www.ifi.unizh.ch/ifiadmin/staff/rofrei/DA/DA_Arbeiten_2002/Stock_Mike.pdf

[45] S. Ziemer. *An Architecture for Web Applications.* DIF 8914 Distributed Information Systems. Nov 2002.

[46] Microsoft. *About the WebService Behavior.* URL: http://msdn.microsoft.com/workshop/author/webservice/overview.asp

[47] M. Secrist. *Portalization, a guide to developing portal ready applications.* Nov 2002. URL: http://devresource.hp.com/drc/technical_papers/portal_app.pdf

[48] Y. P. Shan and R. H. Earle, *Enterprise computing with objects from client/server to the internet.* Addison Wesley, Dec. 1997.

[49] Y, Ding, D. Fensel and H. Stork. *The Semantic Web: from Concept to Percept.* OGAI, 2003

[50] T. Haspels. *Implementing Content Management Systems for the Web in the Nonprofit Sector: A Realistic Approach*. Dec 2003. URL: http://people.mills.edu/thaspels/Proposal/Research.htm

[51] B. Doan, M, Beigbeder, J. Girardot, P. Jaillon. Using Metadata to Improve Organization and information Retrieval on the WWW. WebNet98, Rouen, Nov 1998.

[52] Yahoo Bookmarks. URL: http://bookmarks.yahoo.com

[53] Spurl.Net. URL: http://www.spurl.net

[54] MyBookmarks. URL: http://www.mybookmarks.com

[55] Backflip. URL: http://www.backflip.com