Department of Computer Science and Engineering
The Chinese University of Hong Kong

Final Year Project Final Report
2001~2002

LYU 0101
Wireless Digital Information System

Supervisor
Prof. Michael Lyu

Group member
Lam Yee Gordon
Yeung Kam Wah

Prepared by Yeung Kam Wah

Abstract

Nowadays the mobile device and wireless communication develop rapidly. The business importance of web-enable phones and PDAs become higher and higher. To go with the mobile wireless trend, we propose our project --- Wireless Digital Information system.

Wireless Digital Information system is a generic system for providing information to PDA users. Information are input in the server in form of XML data, The system provide search function and support multimedia content.

Table of Content

Chapter 1 Introduction P.5

Chapter 2 Design Criteria P.6

Chapter 3 System Overview P.8
3.1 Overall Design P.8
3.2 Network P.8

Chapter 4 Client Implementation P.9
4.1 Platform and Tools P.9
4.2 Design P.9
4.3 Main Window P.10
4.4 Image Window P.12
4.5 Video Window P.12
4.6 Customized HTML event on the HTML viewer P.13
4.7 Difficulty P.14

Chapter 5 Server Implementation P.15
5.1 XML and XSL Pre-processor P.15
5.2 Search Engine P.19
5.3 TCP Server P.21
5.4 Web Server P.22
5.5 Video Server P.22

Chapter 6 Communication Format P.23
6.1 Basic Format P.23
6.2 Client Request P.23
6.3 Server reply P.24

Chapter 7 User Guide for Client P.26
7.1 Connect and Login P.26
7.2 “File” Menu P.27
7.3 “View” Menu P.27
7.4 “Movie” Menu P.27
7.5 Searching using the Image Window P.28

Chapter 8 User Guide for Server Administrator P.29
8.1 Match XML file with XSL file P.29
8.2 Program Usage P.31
8.3 Provide searchable image to user P.31

Chapter 9 Algorithm used P.33
9.1 Conversion of XML data into Relational Database P.33
9.2 Search Engine P.35
9.3 NN-search P.36

Chapter 10 XML Related Technology P.45
10.1 XML P.45
10.2 XSL and XSLT P.53
10.3 XHTML and XHTML base: P.59

Chapter 11 Review on Trial Application P.63
11.1 Building our First Application at Pocket PC P.63
11.2 Architecture of First Application at Pocket PC P.65
11.3 Bluetooth and 3G technology P.65
11.4 Problem in our First Application P.69

Chapter 12 Conclusion and Personal Contribution on Project P.70
12.1 Major responsibility on the project P.70
12. 2 TimeLine P.70
12.3 Conclusion on the project 72 P.72

Chapter 13 Reference P.73

Chapter 1 Introduction

Nowadays the mobile device and wireless communication develop rapidly. Handheld mobile devices with access to the Internet and other network application are exploding. The business importance of web-enable phones and PDAs become higher and higher. Moreover, new hardware products such as the Tablet PC and PDA/phone combinations are start to push out to the market. Research also indicates that by 2002 there will be over 1 billion mobile phone owners globally with Internet access. Wireless LANs, Bluetooth, 802.11 and other wireless technologies are rapidly evolving.

With powerful handheld device and sufficient wireless bandwidth, we can predict there is a high demand on information and services provide for mobile device. IBM estimates the overall (both carrier and enterprise classes) market for mobile services alone should equal $30.5 billion by 2003. To go with the mobile wireless trend, we propose our project --- Wireless Digital Information System on PDA.

Wireless Digital Information System

Our project is to build a generic system for provide information to PDA user. For supporting different format of the different type of information, XML is chosen as input data format. Server provides search function on the XML data and formats the result by a set of XSL files. Client provides an interactive interface to user and also supports multi-media content.

This system can be suitable for many applications, especially suitable for location sensitive content. Below are some of possible applications in our generic system.

- Shopping guild in a Shopping Mall

  When users come to a shopping mall, they can search for the information about the shopping hall. The system can provide information such as promoting events occurring, description of the shops, what products of the shops sell which and what sales or promotions are running in the shopping hall. This let the users can locate where they want to go before walk around in the shopping mall.

- Information search in an Exhibition

  Usually the exhibition center is so large and people are not easy to find the counter want to go. With our system, people can search the location they want to go. The system can provide the business nature of the companies, products the companies provide or demonstrate, etc. This can help the busy businessmen save valuable times.

- Showing vacancy in a Car Park

  Using our system, driver can connect to the local server when they go in a car park. They can know where has a space in the car park by using our system. This decreases the time for drivers drive their cars around and find a vacancy,

- Digital video library client

  Except location sensitive usage, our system can also in a more general way. User can connect to a video library, than can search for news and other video content in anywhere they want

- Other usage

  The usage of the system is not limited, server administrators can provide their own content, and to satisfy user's need.

With Wireless Digital Information System on PDA, providing information to PDA users becomes easier. PDA user can also use a single client to connect to different servers, this reduces the inconvenience of the installing different client for different systems, and saves limited memory resource. In the following chapters, we will describe the architecture of our system, user guide and also describe the algorithms we used in the system.

Chapter 2 Design Criteria

Our main targets of our project have three: (1) Provide an easy, effective, feasible, and extensible way for information provider (IPr) to prepare their digital information (e.g. video, image, text), so that we can present their information through PDA with wireless to their customer. (2) Provide efficient and effective searching capability within the given digital information, so that IPr's customer can find their information in an easy way. (3) Provide a PDA client, so that its modules can give IPr to provide as rich as information they can.

Provide an easy, effective, feasible, and extensible way for IPr

To achieve our target (1), we choose XML as our primary data format and XSL as the translation language to convert XML data into information. Reasons for use XML with XSL are obvious. (a) XML can provide a standard, easy, feasible and extensible way for IPr to prepare information. (b) XML is in structural text form, so we can easily handle it. (c) XSL helps to separate the concern on design of the data format and the presentation of data. From (c), there are advantages like, (i) any changes in the Front end (i.e. change in the client application modules, or change in the presentation of information) will not affect the original XML data, and the only change will be on XSL. (ii) IPr's can have a more feasible way to design their presentation of data depend on the client application modules that we provided by XSL. Detail materials on XML and XSL will be in the later chapter "Background"

But there is still problem on using XML data. As XML data is just a page of structural text, although XML data can be design, understand and extend easily, there are not yet any standard and efficient mechanism for managing of XML data format if we use XML as data storage, this can be easily know when we compare the using the relational Database table and XML structure to store the data. Relational Database is well develop today and provides us many useful services, e.g. building different kinds of indexing on data, concurrency control, recoverability of data, etc. Thus, if we can some how have method to convert XML data into Relational Database, then it can help us to solve the problem. In our system, we will break down XML data into "tree node tuples", so that beside the traditional depth first traveling in XML data, we can do breath first traveling or build other tree structure in the Relational Database for speed up our access to our XML data. We discuss two algorithms to do the conversion between the XML and the relational Database tuples in the later chapter "Algorithms"

Provide efficient and effective searching capability

To achieve our target (2), as an academic project, we don't want to use existing tool like "Microsoft Index Server" as our Search Engine, we plan to do our own Search Engine. Building a searching engine may involve techniques like text analysis, modeling, indexing and searching, ranking, relevance feedback, etc. Detail description of our Search Engine will be in the later chapter "Server Implementation.

Allow presentation of information as rich as possible on PDA client

To achieve our target (3), we implement a PDA Client on Pocket PC, which involves three main modules; they are HTML window, Video window, and Map window. We design these three modules have several reasons. (a) HTML is nowadays most common media for expressing information, and it can be easily transform out from the XML data and XSLT. (b) Video is an important media, which can express information more than just text and image in HTML. (c) Map is a useful media, which give an alternative visual functionality on the location of the information that client wants, it enhance the power of expressing information when compare with just use HTML page and hyperlinks.

Chapter 3 System Overview

In this chapter, we will discuss the structure of our system, and the general idea of data flow from the source data to the PDA client.

3.1 Overall Design

Our system's design is base on convenient server/client model with the thin client approach. The server side is responsible for the information preprocessing, data storage, search function and also formatting result to client. But client only responsible for displaying multimedia content, and send request to server respect to user action. The diagram below shows the system architecture.

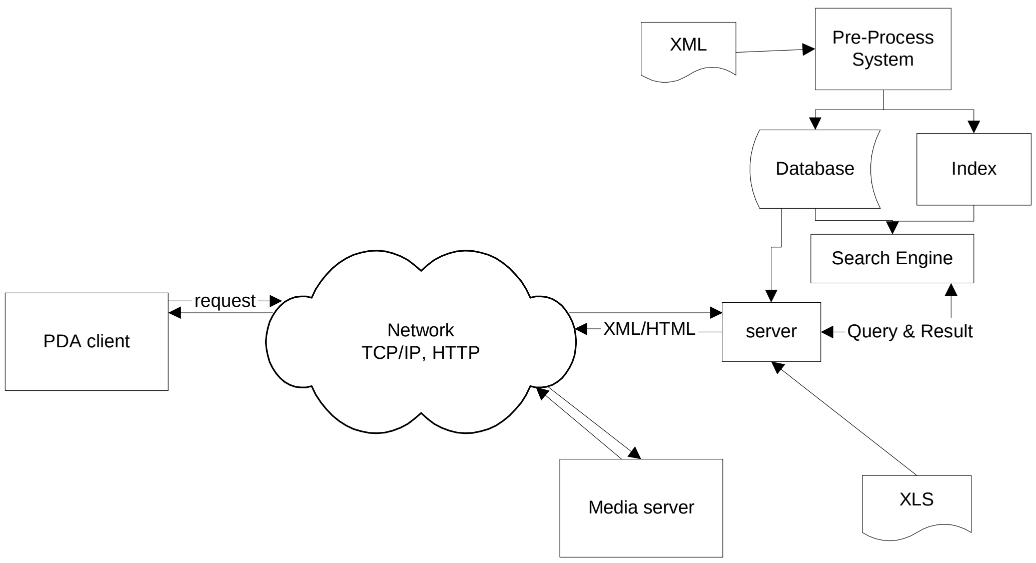


Fig 3.1 Architecture of Wireless Digital Information System

Input of the system is set of XML data and a corresponding set of XSL using to format the data. The XML is passing the pre-process system to transform the format to store in the database, and index for the data is built. When client send a request to server, the server perform a search operation to obtain the result content needed. Then the server formats the result using the XSL and then sends back to client. This part of server only handle the text base content, the multimedia content is handle by a separate media server.

3.2 Network

The protocol we select for the communication between server and clients is TCP/IP. TCP/IP is used because it is the most widely used network protocol and will still popular in the coming days. Most of the existing or coming network technologies, including BlueTooth, Wireless LAN and also 3G, have sufficient support on TCP/IP. These ensure our system can be installed in any network environment nowadays and in the future.

Chapter 4 Client Side Implementation

In this chapter, we will discuss the design of our Pocket PC client, how is being implemented, and the tools and the technique we used in implementation.

4.1 Platform and Tools

Pocket PC

Pocket PC is the platform we selected to build our PDA client. Compare with another popular PDA OS, Palm, Pocket PC PDA have higher capability on computation power, network device, and also multimedia content. Palm OS is target for PDA provide basic functions, such as phone book, and Pocket PC target for PDA use as a hand-size PC. We choose Pocket PC because we want our client can support multimedia content.

Embedded Visual C++

Microsoft provides two programming IDE for Window CE and Pocket PC. They are Embedded Visual C++ (EVC) and Embedded Visual Basic (EVB). They are similar to The Visual C++ and Visual Basic on the Desktop Computer. Embedded Visual Basic provides a more visualize way of to build interactive GUI, and suitable for rapid development and testing, but we use Embedded Visual C++ because it provides more low-level programming and access all the low-level system call. Moreover, the development environment nowadays on Pocket PC is not well developed and most of the resources are using C++ only. This makes us have more feasibility to develop features not provided by EVB.

Win 32

Similar to Desktop Windows Programming using Visual C++, an application can be developed by using traditional Win32 programming or Microsoft Fundamental Class (MFC). Win32 is the traditional way for window programming and it is base on message queues, and MFC is a OO-base API and abstract all the Win32 API behind.

We using Win32 programming as most of resources about Pocket PC we find are Win32 base. Also, similar to using Embedded Visual C++ instead of Embedded Visual Basic, using win32 more feasibility on control the system component such as the HTML viewer and Video viewer.

4.2 Design

Client side is a program run on Pocket PC using Win32 API, it provide three different windows for different kind content and acts as bridge between users and backend server. The three different windows are the Main Window, the Video Window and Image Window, and these windows are controlled by a control component.

Control component responsible for communication with the server and control the cooperation of the different windows. The Main Window basically is an HTML viewer. It is the main interface of the client program. Most of user interaction and information presentation is using this window. The image window is an image viewer with highlighted points on it. It is good for map like image using highlight point to indicate locations. Image window can also be an input of query, this mean user can be drag a rectangular area as a query input. The video window is used to present video content to user. We will discuss different window in more detail coming parts. Below is a diagram showing the relation of different components.

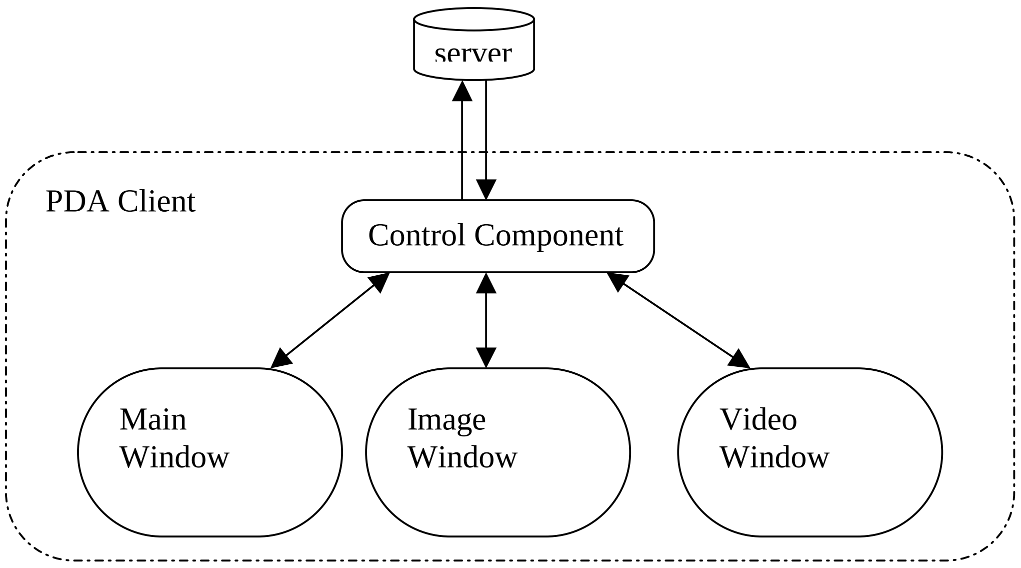


Fig 4.1 Different components in the Pocket PC client

4.3 Main Window

The main window is the main user interface of the Pocket PC client. It responsible to display text and image content and receive user respond. User respond will pass to and handle by the control component. Use the HTML viewer as main interface because of the benefit provided by the feasibility presenting information of HTML. With HTML, the user interface can be adaptive change by server to satisfy the kind of information display.

The HTML viewer used a Dynamic-link Library provided by Pocket Internet Explorer (Pocket IE). As Pocket IE is pre-installed with the Windows CE in every Pocket PC PDA, so it is a system provided library, and no special installation required. The HTML viewer fully supports on standard HTML 2.0. When server request event generated, such as user click on a <a></a> link or a bottom of <form></form>, will pass to its parent windows (which is the control component in our client). So we use this feature to simulate and customize the respond on button event and text field event. In the coming part, there is section to show how to customize the <a></a> and <form></form> event.

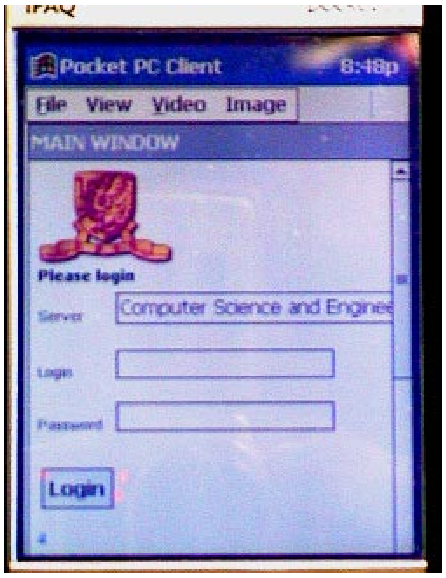


Fig 4.2 The Main Window

These are some system functions that related to the HTML viewer.

LoadLibrary
Prototype
    HINSTANCE LoadLibrary(LPCTSTR lpLibFileName);
Usage
    Load the required DLL to memory. For HTML viewer, corresponding DLL file is “htmlview.dll”.

InitHTMLControl
Prototype
    BOOL InitHTMLControl(HINSTANCE hinst);
Usage
    Do the required initialization for the HTML viewer. It is necessary for execute this function before execute other functions related to HTML viewer.

SendMessage
Prototype
    void SendMessage (g_hwndMain, WM_PUTTEXT, NULL, (LPARAM) lpBufferW);
Usage
    Send a message to a window. For HTML viewer, there are three message related to the display of HTML content: WM_SETTEXT, WM_ADDTEXT, WM_ADDTEXTW, which is used to set or append the HTML content displaying on the HTML viewer. In the LPARAM field of the message is the pointer the

When there is a server event that generate HTML viewer will send a WM_NOTIFY message to the parent window, LPARAM field will be pointer to a NM_HTMLVIEW struct. The NM_HTMLVIEW struct contain a field that indicates the kind of message. These are messages that HTML viewer will send to parent window.

NM_INLINE_IMAGE:

This message request the parent window that load the required image, the location of the image is given by the target field of the NM_HTMLVIEW struct.

NM_HOTSPOT:

This message send to parent window when user generate a server request event on the HTML in main window, server request event is include user click on a link (<a></a>), or the submit button of a form (<form></form>).

4.4 Image Window

Image window that display an image with or without highlight point, it is designed for map like image using highlight point to indicate locations. Image window can also be an input of query, this mean user can be drag a rectangular area as a query input. This window only uses some basic Windows API and a list of highlight points are kept in a data structure.

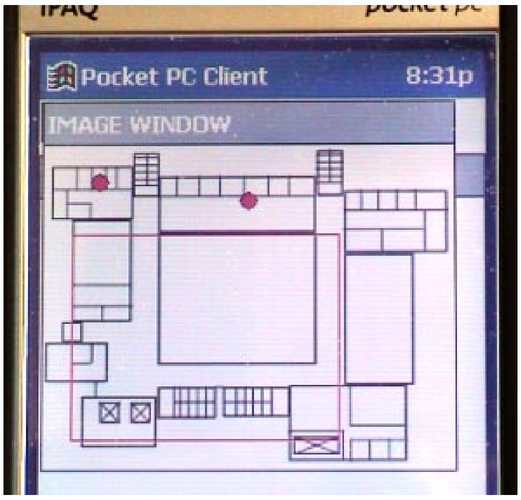


Fig 4.2 The Image Windows

4.5 Video Window

The video window is the interface to provide video/audio content on the Pocket PC client. It is responsible for video or audio playing.

The video viewer used a COM object provided by Windows Media Player for Win CE. Same as Pocket IE, Windows Media Player is pre-installed with the Windows CE in every Pocket PC PDA, so it is a system provided library also, and no special installation required. By the function provided by the COM library, the client can support all media format support that support by Windows Media Player and it also support streaming of multimedia content.

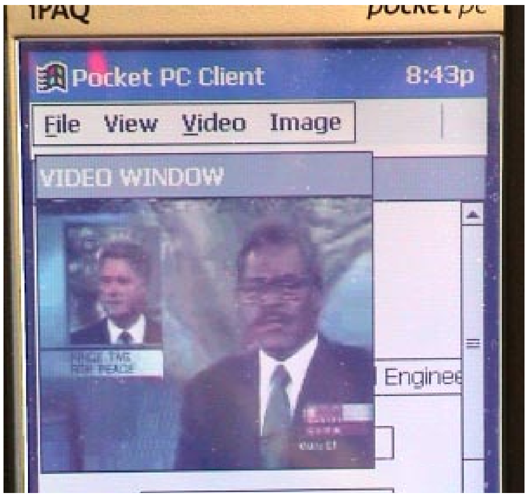


Fig 4.3 The Video Window

There is a series of function that used to control the Video viewer. The control of the video viewer is through a set of interface.

- ?? IGraphBuilder *pGB;
- ?? IMediaControl *pMC;
- ?? IMediaEventEx *pME;
- ?? IVideoWindow *pVW;
- ?? IBasicAudio *pBA;
- ?? IBasicVideo *pBV;
- ?? IMediaSeeking *pMS;

And each interfaces of the video viewer are initialized by following function.

```
CoCreateInstance(CLSID_FilterGraph, NULL, CLSCTX_INPROC_SERVER,
IID_IGraphBuilder, (void **)&pGB);
pGB->RenderFile(wFile, NULL);
pGB->QueryInterface(IID_IMediaControl, (void **)&pMC);
pGB->QueryInterface(IID_IMediaEventEx, (void **)&pME);
pGB->QueryInterface(IID_IMediaSeeking, (void **)&pMS);
pGB->QueryInterface(IID_IVideoWindow, (void **)&pVW);
pGB->QueryInterface(IID_IBasicVideo, (void **)&pBV);
pGB->QueryInterface(IID_IBasicAudio, (void **)&pBA);
```

After set the size and location of the video window, the video playing can be controlled by following function.

```
pMC->Run();
pMC->Stop();
```

4.6 Customized HTML event on the HTML viewer

As we are using the HTML viewer as the main interface, so we need to customize the HTML event to simulate user events. We customize the HREF field of the <a></a> tag and <form></form> tag to satisfy our need. Below are the tables show how we customize the HREF of the <a></a> tag and <form></form> tag.

Format	Action
Request?format=<format_name>& main=<XML file>&map=<XML file>&video=<XML file>	Request the server format the required XML file and format it.
Video <URL>	Play the video placed at URL in the Video viewer
Video Stop	Stop the video playing on the
<URL>	Send to server and display server respond on HTML viewer.

Table 4.1 Customize the HREF field of the <a></a> tag.

Format	Action
Login <URL>?name="<username>" && password = "<password>"	Connect and login to the server specified at <URL> with <username> and password
Search <URL>? keyword = "<keyword>"	Search item <keyword> on the <URL> server
Others	Send to server directly, and display the server respond on HTML viewer.

Table 4.2 Customize the HREF field of the <form></form> tag.

4.7 Difficulty

Lack of learning resource

Compare with programming in desktop windows or palm OS environment. Resource and documentation on Windows CE are not so clear. Although most of the documentation can be refer to the documentation of desktop Windows, not exactly the same set of APIs are supported in Windows CE and desktop windows, so extra time is needed to try the APIs to build the PDA client.

XML library in Pocket PC

Although Pocket IE has the capability to handle XML data, the library is not open to other program. We need to handle the XML pausing by our own code. This makes the program more complex and less capability to make change.

Chapter 5 Server Implementation

In server side, there are five main modules, they are:
(5.1) XML and XSL Pre-processor – for preprocessing of XML and XSL data
(5.2) Search Engine – involving analysis on the XML data
(5.3) TCP Server – responsible for the communication with PDA client
(5.4) Web server (images) – storing the entire image
(5.5) Video server (MMS) – storing and steaming all video

In the following page, we will explain more on these five modules.

5.1 XML and XSL Pre-processor

This module involves the preprocessing of the XML and XSL data. Fig 5.1 help us to explain the whole processing.

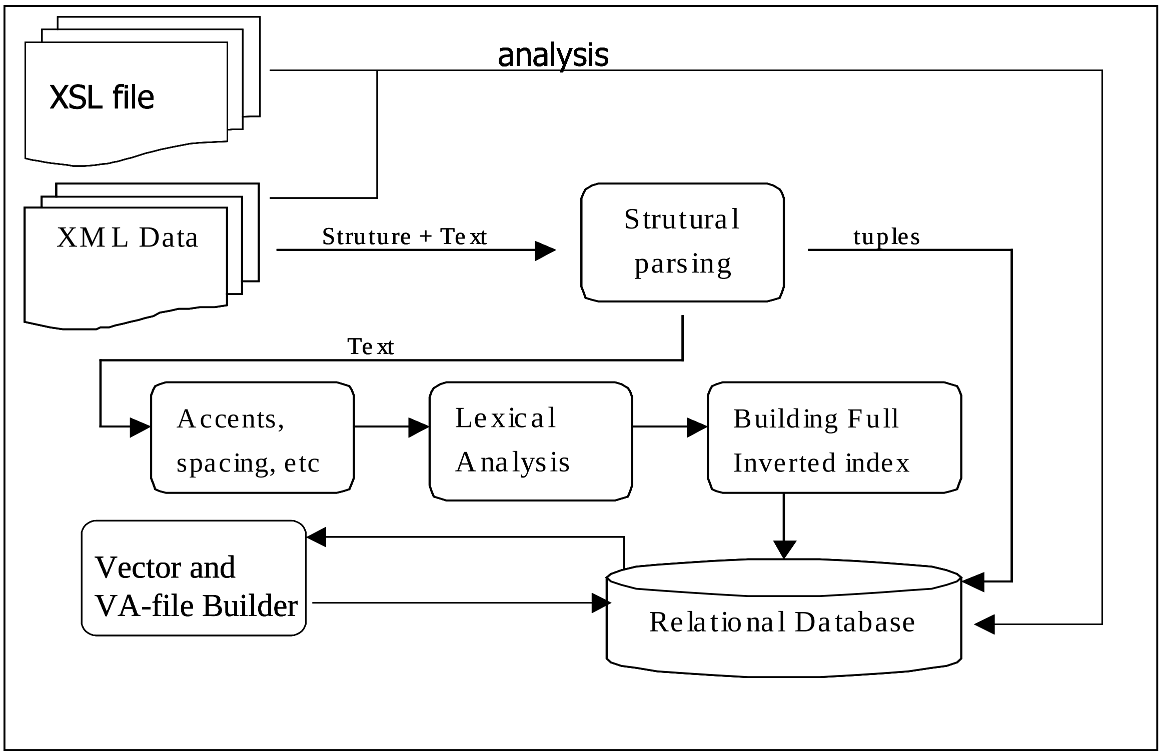


Fig 5.1 Process flow of the pre-process system

5.1.1 Handling of XSL file

First, before we talk about the handling of the XSL file, we need to state out some relation between the XML file and XSL file. One XML document can have several set of XSL file to transform it. Each XSL file in the XSL set will correspond to one component for the PDA client. For example, we can have an XSL set which contains two XSL file, one XSL file for transforming an XML document to HTML (for the HTML window), and one XSL file for transforming another XML document to Map definition (for the Map window, and the format will be specified in Chapter 6).

Therefore, in our Database, we first have the following three schemas for the XML and XSL file, they are:

i.) **XMLNAME**

ID	NAME	NODEID

This table will store up the name of the XML file, the NODEID is the root node's id in Table XMLTREE (will be discuss later).

ii.) **XSLNAME**

ID	NAME

This table will store up the name of the XSL file.

iii.) **XSLSETNAME**

ID	**XSLSETNAME**

This table will store up the XSL set name

iv.)**XSLSET**

XSLSETID	XSLNAMEID

XSLSETID is the ID from the Table XSLSETNAME. This table will store up the set of the XSL file that in that XSLSET.

v.) **XSLSETCONTAINER**

XSLSETID	SID	XMLNAMEID

This table will store up the XML set for each XSL set. SID is the set id of the XML set, and XMLNAMEID is the ID in Table XMLNAME

Just says an example that, give one XSL set {"HTML1.xsl","MAP 1.xsl"} name "TRANSLATION1", and we have two XML set {"HTML11.xml", "MAP11.xml"} and {"HTML12.xml", "MAP12.xml"} for that XSL set. Then, we will have the following tuples:
i.) XMLNAME
(1, HTML11.xml, #)
(2, MAP11.xml, #)
(3, HTML12.xml, #)
(4, MAP 12.xml, #)
ii.) XSLNAME
(1, HTML1.xsl)
(2, MAP1.xsl)
iii.) XSLSETNAME
(1, TRANSLATION1)
iv.) XSLSET
(1, 1)
(1, 2)
v.) XSLSETCONTAINER
(1, 1, 1)
(1, 1, 2)
(1, 2, 3)
(1, 2, 4)

Then, if there is a new XSL set {“MAP2.xsl”} named “TRANSLATION2”, one XML file set {“MAP11.xml”} for this XSL set. Then, we will have the following more tuples:
ii.) XSLNAME
(3, MAP2.xsl)
iii.) XSLSETNAME
(2,TRANSLATION2)
iv.) XSLSET
(2, 3)
v.) XSLSETCONTAINER
(2, 1, 2)

5.1.2 Handling of XML file

The Handling of XML will be much complicate; it involves two major paths of handling, 5.12a.) Conversion of XML data into Relational tuples. 5.12b.) Text analysis and Building the Full inverted index.

5.1.2a Conversion of the XML data into Relational tuples

The detail method of the conversion and the reasoning of using it will be written in later Chapter “Algorithm”. Now we just state out the Relational Schema that for storing XML data:

XMLTREE:

PARENTID	**NODEID**	TYPE	VALUE

TEXT:

ID	**SID**	CONTENT

Each XMLTREE tuple will contain the PARENTID, which is for pointing out who is the parent node of this XML node, and the NODEID, which is the unique ID of that node.

Both NODEID and PARENTID will be on fly generate when parsing the structure of the XML data. TYPE will be store up the type of the node, and the value mapping between the type and the number are state in the below table

Type	Number
tag	0
attribute	1
attribute value	2
text	3
comment	4

Then VAULE will be i.) tag name, ii.) attribute name, iii.) attribute value, iv) text’s ID, v.) comments.
For the type of “text”, we just store up the value of text’s ID, which is the ID of the table “TEXT”. We need to do this because of the value of the type

“text” may exist the maximum length of the VARCHAR that can be declared in our Database, so another table “TEXT” is need.
As you see, there is SID in the table “TEXT’, which is the segment ID of a text. The use of the SID is for the separating of long text. Let’s say if we have a text, which have length 5112, then this text need to be separated into 5 tuples with SID 1,2,3,4,5 in the table “TEXT”, given that the max length of the VARCHAR that can be declared is 1024, and each segment will be put in the CONTENT

5.1.2b Text analysis and Building the Full Inverted index

This path will involve the text analysis of text of the XML file and finally building the Lexicon and Full Inverted index.

Text analysis is for eliminating useless character, e.g. space. And the result of Text analysis will be the Lexicon that we want. Detail will be discussed later Chapter “Algorithms”.
For the text analysis, we have done the following things:

i.) Handling of UNIQUE code

We make analysis on the UNIQUE coding, and find out the following relations.

a.) Symbols: 0 – 47, 58 – 64, 123 – 191, 8204 – 8260, 12288 – 12320, 61444 – 61489, 63232 – 63260, 65072 – 65131, 65281 – 65295, 65306 – 65312, 65371 – 65381, 65504 - 65518
b.) Foreign words: 0 – 4601, 65152 – 65276, 65313 – 65338, 65345 – 65370, 64288 - 64509
c.) Number: 65296 - 65305
d.) Chinese and Japanese: 19968 – 40869, 59413 – 59492, 63744 – 64045
e.) Japanese: 12353 – 12976, 65382 – 65438
f.) Korean: 44032 – 55203

We will flow away all the Symbols. Symbols are involving space, tag, accent, Mathematic symbol, etc. Moreover, the remaining will the Number and Words. Handling of these separate into two main groups of words, one group is foreign words and Number; another group is Chinese, Japanese, and Korean. They need different handling because of the first group is separate by space, accent, etc, but the second group need to separate one character by one character. Just say an example, “I am 中大 student” will be separated to “I”, “am”, “中”, “大”, “student”.

ii.) Stopwords removal

We also have our own stop word list, which is for removal of useless word like “is”, “are”, “very”, “there”, “but”, “by”, etc. In our project, we only work out the stop word list of English.

After the Text analysis, Lexicon comes out at the result. Lexicon will be used in the Search Engine part as the index term. The following Table is the schema for the Lexicon:

LEXICON:

ID	Name

This LEXICON table will store up all the important word within the collection set of XML document.

For the full inverted index, we will be on flying built when doing the Text Analysis.

The following Table is the schema for the full inverted index.

INVINDEX:

LEXICONID	XMLNAMEID	POS

Full inverted index is the indexing, which not only remember which XML document contains which words, but also remember the position of those words appear in the XML document. We build this full inverted index for the following reasons:

i.) We can have a better ranking, e.g. search for “林怡”, I prefer to have “林怡” in search result, rather than “林保怡”.

ii.) It can help in searching a whole phase, e.g. “一勞永逸”

5.2 Search Engine

Some preprocessing works on the Search Engine part have been done in the XML preprocessor, they are Text analysis which for generate index terms and Building of Full Inverted Index for ranking. In this module, there are still separate into two main parts, 5.21.) Building Vector Model and VA file. 5.22.) Ranking

5.2.1 Building Vector Model and VA file

Vector Model is a model for building vector for each document as the basic unit for similarity search. Detail explanation for Modeling of document will be explained further in later Chapter “Algorithm”, but here will be explain a little on what is Vector Model.

In Vector model, within one document, a weight is given for each index term (the lexicon in the before chapter), so the weight of all index terms forms a vector for that document. Weight is calculated by the multiply of tf factor – term frequency and idf factor – inverse document frequency. For tf factor, it measures the importance of an index terms within one document, i.e. if an index term appears many time in a document, then the value of tf factor of this index term will be high in this document. On other hand, idf factor is for the measurement of the importance of an index terms in all the document, i.e. if an index term appears only in a small set of document, then the idf of this index term will be high in those documents. With this vector, we can do a similarity search between the query and all documents. The measures of similarity in our project will L1 distance function, so normalization is need for the vectors.

Here is the schema for us to store up the vector of each document:
WVECTOR

XMLNAMEID	LEXICONID	WEIGHT

Therefore, with the XMLNAMEID, we can get the vector out.

Then, before talking about VA-file, we first talk about Nearest-Neighbor search (NN search). NN search can specify the number of the nearest result that we want to obtain. In our case, we will use NN search for take out at most 50 most relevant result pages. Just take an example, if we search for "Search Engine" in our xml documents, there may be more than thousands of result xml document, but if we use the NN search, we can specify that we only want the first 50 result. You can see there can be great different of implementation between you find all the search result and you just find first 50 result. VA-file is one of method for NN search, and we choose VA-file for cut down the search page, this is important as the PDA cannot handle too much result page at a time.

There is many NN-search algorithms, which will be discuss in later Chapter "Algorithm", but VA-file suit for our case. VA-file outperforms other Tree structure algorithm in high dimension space in term of IO page, number of Vector calculation, CPU time in searching and CPU time on construction. Main idea of VA-file is come from the "dimensional curse" of the existing MBR algorithm. MBR algorithm will recursively form some Hyper Sphere or Hyper Cube for partition of data, so the Hyper Sphere or Hyper Cube will help to prune out the searching space in NN-search.
But the problem of MBR is that when the dimension getting higher, but the data size can not catch the growing of it, then pruning power of the Hyper Sphere and Hyper Cube will largely decrease. You can just imagine that the case of ten point in 2D space and then point in 3D space, partitioning points in 2D space will be much efficient than 3D space. Therefore, VA-file suit for our project as the dimension of Vector of each XML document can be up to hundred or thousand.

In many experiment, just simple linear search will work better than the MBR algorithm, it is because the run- time is not just depend on the vector calculation, but also on the IO request. As MBR algorithm will build a tree structure for filtering, but that tree structure is not that small (as vector to represent a Hyper Sphere or Hyper Cube is as large as a data vector) Traveling in that tree structure will need to jump to an unpredicted IO-page, which will slow down the whole processing. Despite this, the computation for filtering that makes use of Hyper Sphere or Hyper Cube will just like the computation among data points, so we can consider that the Hyper Sphere and Hyper Cube are new data points. Therefore, VA-file uses filtering algorithm on the linear searching.

In the following is the table Schema for the VA-file:
VABOUND

LEXICONID	POS	VALUE

VAFILEI

VALUE

VAFILE

VALUE

VABOUND is for the storing of the filtering information, main idea is to do an non-linear truncation in each dimension of the data point for filtering.

VAFILEI is for the storing of a list of XMLNAMEID that participate in the VA-file.
VAFILE is for the storing the truncated vector for each XML document.

The Method of Building VA file will be discussed in the later Chapter "Algorithm".

5.2.2 Ranking

Ranking of the result page is important part, which finds out the relevant XML file content. Basic ranking will be just measure the similarity between the vector of the query and XML documents, so in our case, the nearer distance will have higher ranking. After make use of the NN-search by VA-file, we can specify the N next most relevant xml document given xml document in the bottom ranking in the search result page.

In addition, we make use of the Full Inverted Index to have a better ranking by the position information of the index term inside an xml document. We will rank the page higher if the query words come near together within an xml document. Just give an example, if you search for "Search Engine", I will rank the XML documents with exactly the word phase "Search Engine" higher than those XML documents just contain "Search" and "Engine" separately. We will do this ranking method just after the similarity search.

5.3 TCP Server

Main functionality of the TCP Server is for reply the request of the PDA client.
The main request from the PDA client is specified in the later Chapter "Communication format". There is three main request from the PDA client, i.) keyword search, ii.) map search, iii.) XSL request.

i.) Keyword search – in this case, we transform the query into vector, and then do the NN-search and ranking the result by the Full Inverted Index, lastly, we need to locate the XSL set for the entire relevant documents and generate a result page for it.

ii.) Map search – in this case, the server will receive an rectangle co-ordinates and the filename of the xml for a map definition. As each highlight point in the map definition will be related to some information, so the server will generate all those result in the result page.

iii.) XSL request – XSL request will consist of the name of the XSL set and the list of XML filename that need to be translated in XSL set. The result will be directly display in the related PDA client windows.

5.4 Web server

Any web server will be suit for our project, but due to our video server is using the Microsoft MMS (Multi-media server), so we use the Microsoft IIS server.

5.5 Video server

We use the Microsoft MMS server for video server. MMS uses the UDP transport layer protocol. MMS can use Multicast so that a large number of connections can be established without any extra work and bandwidth on the server. The use of MMS requires that certain ports be open on the client's firewall. For a powerful enough computer with a small number of connections, both MMS and Window Media Encoder may run on the same machine.

Chapter 6 Communication Format

In this chapter, we will describe the communication format between the client and server. The communication of the data is using text base and using XML format, this allow people to build their own client on different platform but support by the same server in our system.

6.1 Basic Format

As the communication between server and clients are stream-base (TCP/IP), the communication stream is divided in to messages. Each message is base on the basic format below.

Length	Content

Fig 6.1 Basic Format of a message

Length is the 4-byte integer in network order represents the length of content in number of bytes and the content is a Unicode string usually in XML format.

The message flow is like this, each time a client sends a request to server, server handle the requests, and return a set of reply to tell client how to update different components.

6.2 Client Request

1. **Login message**

   Format:
   ```
   <login>
       <userid>( user name )</userid>
       <password>( password )</password>
   </login>
   ```

   Description:

   This is the first message that the clients send to server after the connection is established. It contains the information to identify the user.

2. **Default page message**

   Format:
   ```
   </defaultPage>
   ```

   Description:

   This messages that request the default page of this server. It used as the “home” key of a web browser.

3. **Request page**

   Format:
   ```
   <request>
   ```

```
        <link xsl_set="(xls_set_name)" >
            <main>(name of XML file requested)</main>
            <map>(name of XML file requested)</map>
            <video>(name of XML file requested)</video>
        </link>
    </request>
```

Description:
This message is used to request the server to format the required XML files give reply. This message usually generate by user kick on interactive components with control word “link”.

4. **Search message**

Format:

```
<search>
    <keyword>( list of    keyword ) </keyword>
    <no> </no>
  </search>

  or

  <search>
    <img>(filename of search image)</img>
    <x1></x1>
    <y1></y1>
    <x2></x2>
    <y2></y2>
  </search>
```

Description:
This requests the server to perform the search function. The first format performs a keyword search and it is generated by user kick on interactive components with control word “search”. The second format performs a search on an area of image, and it is generated by user mark an area in the image window and selects the image search.

6.3 Server reply

1. **Main Window Reply**

Format:

```
<HTML>
    (The HTML code control the display main window)
</HTML>
```

Description:
This message tells the main window update the display.

2. **Map Window Reply**

Format:

```
<map>
            <img>(file name)</img>
            <query_file>(XML filename)</query_file>
            <highlight>
                  <x></x>
                  <y></y>
            </highlight>
            …….
      </map>
```

Description:

This message tells the image window the location of image need to display and the set of highlight points need to highlight.

3. **Movie Window Reply**

Format:

```
    <smil><body>
            <par>
                  <text src="( description of    the video )"/>
            <video src="(URL of the video)" fill="freeze"/>
        </par>
    </body></smil>
```

Description:

This message format tell which video can be play in this stage. The reply is a subset of SMIL, this is because SMIL is a standard to describe the presentation of the content, and make the reply format more scalable and capable with existing and coming systems.

Chapter 7 User Guide for Client

In this section, we will describe how to use the PDA client to obtain information from server.

7.1 Connect and Login

To connect to the server, select to the server you want connect to in the combo box and entry user name and password.

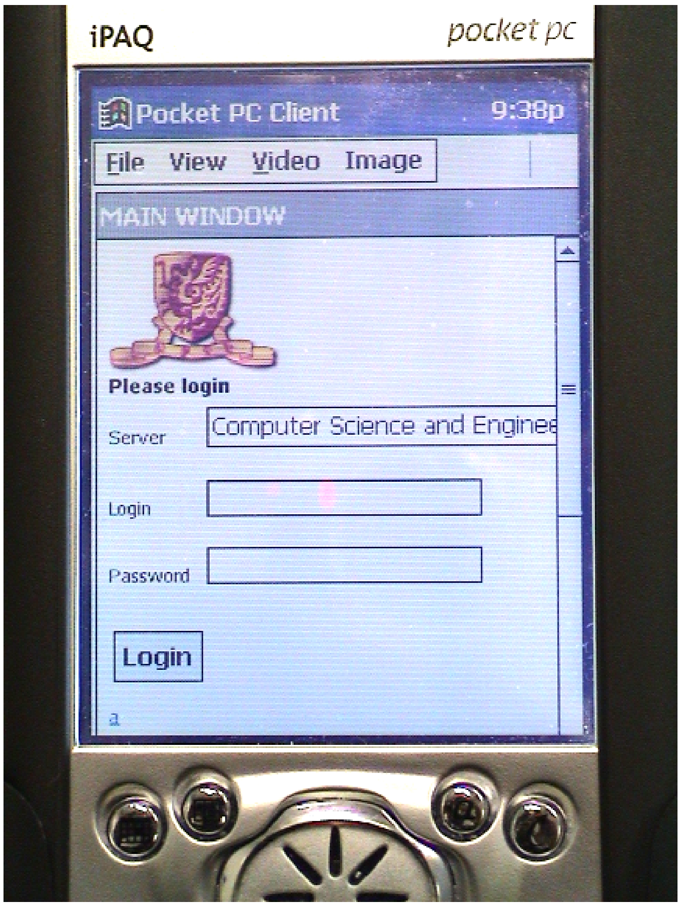


Fig 7.1 The login Page of the PDA Client

If the connection success, the server will send back the default page of server.

7.2. “File” menu

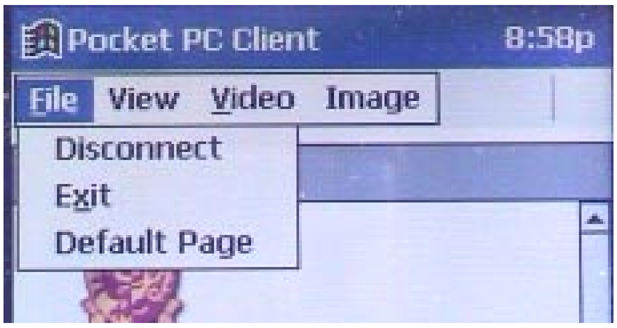


Fig 7.2 The “File” Menu

Default Page

To return to the default page of the connected server, select the “Default page” in the “File” menu.

Disconnect

Select this item to disconnect the selected server

Exit

Exit the client.

7.3. “View” Menu

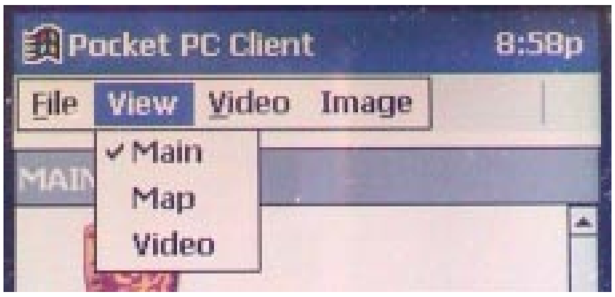


Fig 7.3 The “View” Menu

Main

Toggle the visibility of the Main Window

Map

Toggle the visibility of the Image Window

Movie

Toggle the visibility of the Video Window

7.4 “Movie” Menu

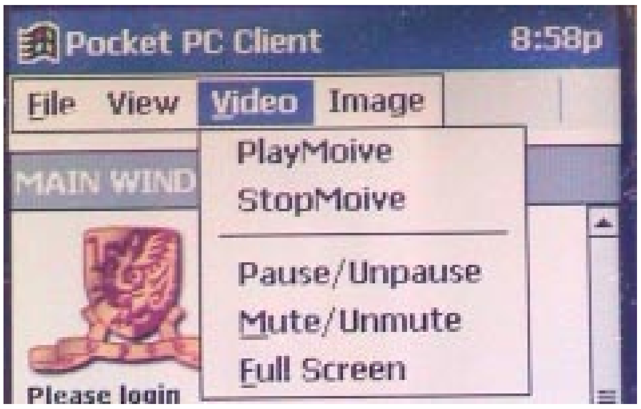


Fig 7.4 The "Video" Menu

Play Movie
    Start the movie in the movie windows

Stop Movie
    Stop the playing movie in the movie windows

Mute/Unmute
    Toggle the mute status of the sound

Pause/Unpause
    Toggle the pause status of the movie

Full Screen
    Play the movie in full screen mode

7.5 Searching using the Image Window

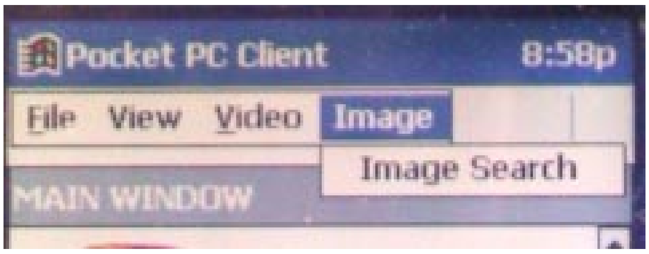


Fig 7.5 Select "Image Search" to query

You can try to search at any time on the image window. Drag a rectangular area in the image window, and then select "Image Search" in "Search" menu (Fig 7.5).   A request will automatically sent to the server.

Chapter 8 User Guide for Server Administrator

In this chapter, we will describe how server administrator can provide information to PDA user.

8.1 Match XML file with XSL file

To format the required XML file to PDA user, the server administrator need to provide related XSL file. Server administrator need to import four files.

First file is the file name list of the entire xml document. Administrator can write down the entire xml file name within a file, e.g.

```
HTML1.xml
VIDEO1.xml
MAP1.xml
……………
```

Fig 8.1 a file that contain file name list of XML

Second file is the file name list of the entire xsl document. Administrator can write down the entire xml file name within a file, e.g.

```
HTML1.xsl
VIDEO1.xsl
MAP1.xsl
…………...
```

Fig 8.2 a file that contain file name list of XSL

Third file is for definition of the XSL set. Administrator can write down the XSL set name and the related XSL file name, blank line will be used for separate each XSL set, e.g.

```
HTML1
HTML1.xsl

VIDEO1
VIDEO1.xsl

MAP1
MAP1.xsl

HTMLVIDEO1
HTML1.xsl
VIDEO1.xsl

VIDEOMAP1
VIDEO1.xsl
MAP1.xsl

…………..
```

Fig 8.3 a file that contain XSL set

Fourth file is for definition of the XML set. Administrator can write down the XSL set name and the XML set, blank line will be used for separate each XML set.

```
HTML1
HTML1.xml

VIDEO1
VIDEO1.xml

MAP1
MAP1.xml

HTMLVIDEO1
HTML1.xml
VIDEO1.xml

………………
```

Fig 8.4 a file that contain XML set

Note that we can only handle the unique name for XML, XSL, XSL set and XML set. Moreover, if we cannot find the translation of an XML file, we will directly output to the PDA client depend on whether it is already the valid format for the PDA client window. For the detail of format about different windows reply, please refer to chapter 6.3.

8.2 Program usage:

There are three main programs, i.) XSMLpreprocessor ii.) VAfile_b iii.) server

i.) Usage:

```
./XSLMLproprocessor    Options
        Options are:
        -a <filename>     load the file of XML file name list and do the XML
  file preprocessing.
    -e <filename>     load the file of XSL file name list
        -i <filename>     load the file of XSL set
        -o <filename>     load the file of XML set
```

ii.) Usage:

```
./VAfile
```

iii.) Usage:

```
./server
```

8.3 Provide searchable image to user

To allow user drag a rectangular area on the image window and perform a search, a XML file is needed to describe a image. This file describes all the possible highlight points on the image and the related XML data file of these XML file. Below is a sample this kind XML file.

```
<image>
        <file>http://www.cse.cuhk.edu.hk/~kwyeung/image/9f.gif</file>

        <highlight>
                <x>30</x>
                <y>20</y>
                <name>Prof. Cai</name>
                <relatedXML>prof_cai.xml</relatedXML>
        </highlight>
        <highlight>
                <x>35</x>
                <y>20</y>
                <name>Prof. Lyu</name>
                <relatedXML>prof_lyu.xml</relatedXML>
        </highlight>
        ………..
</image>
```

Fig 8.5 a sample XML describe possible highlight points in a image

Chapter 9 Algorithm

In the following chapter, we will summarize some algorithm or processing on 9.1) conversion of XML data into Relational Database 9.2) Search Engine 9.3) NN-search

9.1 Conversion of XML data into Relational Database

Main reason for why we want to have conversion between XML data and Relational Database is that we want to make use the well develop Database nowadays to handle XML data.

Except the method that we mention in our project, there is one common way to convert the XML data into Relational Database by the DTD of the XML. Now, we talk about the DTD method first, and then our own method.

9.1.1 DTD method for conversion between XML data and Relational tuples
[XMLDB 1]

Main idea of this method is to have a mapping between the DTD and the Relational Table, so that all the data can be conversion according to that mapping.

$$\begin{tikzcd}\boxed{\text{simplified DTD}} \arrow[r, "\text{mapping}"] & \boxed{\text{relational model}}\end{tikzcd}$$

Simplified DTD means that DTD need to do some simplification before it can convert into relational schema.

Just take an example: we have an DTD below

```
<!ELEMENT author ( name, address )>
<!ATTLIST author id ID REQUIRED>
<!ELEMENT name(firstname?, lastname)>
<!ELEMENT firstname(PCDATA)>
<!ELEMENT lastname(PCDATA)>
<!ELEMENT address ANY >
```

Then we will have schemas:

author

author id	name id	address id

name

name id	firstname id	lastname id

firstname

firstname id	firstname

lastname

lastname id	lastname

address

address id	address

But the about conversion, although it is straight forward, but the table "firstname", "lastname", and "address" is the result of excessive fragmentation, as it just store up one data. So inline technique is used for solving this problem. The following schema is the new one with inline technique:

author

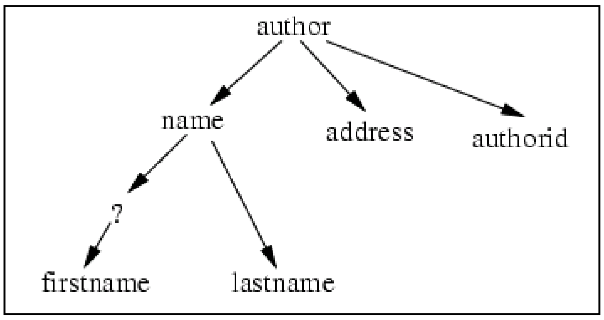
<u>author id</u>	name id	address

name

name id	firstname	lastname

The about example is just a very simple example, as there will be other problem that need to solve like the DTD definition have looping inside it.

This conversion method is just like to break the different type of XML tree node into its own defined table tuples.



9.1.2 Our own conversion between XML data and Relational tuples

As you see in the 9.1.1, there have some problems:

i.) If we have a large DTD definition, then the number of schema formed in the Database will be larger.

ii.) Apart from DTD is used for verify the format of an XML file, its now also need to be involved in the processing of the XML data within the Database.

iii.) The link between the XML tree node points to its child node, as usual, we will make query on that constraint on the child node, so the nested query will need to be make, but this result in inefficient query. Just take an example, we want to find the author with lastname = "Lam", so we first make a query to find out the entire name id with lastname = "Lam", then we use this name id to find which author will have this name id. But this is not efficient as the name id is not the key inside the table author, so searching of which author has this name id will be time wasted. You can imagine that if the constraint on the query is much deeper, nested query will be very inefficient.

From the above observation, we have make improvement on it.

In our project, DTD will just for verify of the XML file, it will not involve in the conversion. Our conversion will also break down the XML file into node, but this time, each node will just store up its parent ID, but not the child ID, this solve the problem of i.), ii.) and iii.). For case i.), as the size of node will be just keep constant (every node will just have at most one parent, but may have many children), so we can keep the entire tree node into one Table. For case iii.), the problem of inefficient in nested query is solved, as the parent ID is the primary key of parent node, so use the same example as problem iii.), after we find out all the parent ID of node with lastname = "Lam", and then we can directly find the author node with that parent ID. And for case ii.), we don't need the DTD related to any processing of XML data with Database.

Here is the table schema for the XML node:

XMLTREE

PARENTID	NODEID	TYPE	VALUE

Where TYPE can be "node name", "attribute name", "attribute value", "text", or "comment". VALUE is the value of the TYPE.

So if you give us the an XML file below:

```
<author id = 1>
        <name>
                <lastname> Lam </lastname>
                <firstname> Yee </ firstname>
        </name>
        <address> HK </address>
</author>
```

It will break down into:

null	1	node name	author
1	2	attribute name	id
2	3	attribute value	1
1	4	node name	name
4	5	node name	lastname
5	6	text	Lam
4	7	node name	lastname
7	8	text	Yee
1	9	node name	address
9	10	text	HK

With this form of table, we can build on primary key is the node ID and secondary key on Parent ID, so that we can speed up either the upward query or downward query.

9.2 Search Engine

In this part, we will explain more on the text analysis, modeling, and relevant feedback.

9.2.1 Text analysis

Main purpose of the text analysis is to find out the index terms. There involve three main step of text analysis; they are Lexical analysis, Elimination of Stopwords, and Stemming.

i.) Lexical analysis -

It is for analysis the digits, hyphens, punctuation marks, case of letters, coding of the text

1.) Number -
Number usually discarded out, but special numbers like year, card number or date may be need special template for identify them.

2.) Hyphens
e.g. treating ‘rule-of-thumb’ and ‘rule of thumb’ identically.

3.) Punctuation
Usually punctuation marks are removed.

4.) Case of letter
Convert all the letter to upper or lower case will help in the identify the index terms, but in some case, Upper case word have more important meaning.

5.) Coding of text
If we document contain other text encoding, e.g. Big5, then special handle is need. It is very complicated if we need to handle document with different language at the same time

ii.) Elimination of Stopwords -

It is for filtering out the word that have very low discrimination values for retrieval purpose, e.g. Noun words frequently carry more semantics than adjectives, adverbs, and verbs.

Stop word list is need in this process, and verify whether a word is stop word or not usually will use the Binary search tree or Hashing.

iii.) Stemming -

It is for removing affixes (i.e., prefixes and suffixes) and allowing the retrieval of documents containing syntactic variations of query terms (e.g. explore, exploring, explorer, exportable), e.g. if you give me {exportable, explorer, exploring}, I will convert them into ‘explore’

Reasons for doing Stemming have two: (1) It improves retrieval performance since it reduces variants of the same root word to a common concept. (2) It also reduces the size of the index terms and indexing structure.

9.2.2 Modeling

There are many different modeling, like Boolean Model, Vector Model, Probabilistic Model, etc. As in our project is due with Vector Model, so here we just explore what Vector Model is.

Vector Model

Vector model makes improving on the Boolean Model by assigning non-binary weights to index terms in queries and docs. So we can compute the degree of similarity between each document and user query, as the result have better ordering of documents.

Measure of Similarity

We can calculate the degree between two vectors as the similarity measure, i.e. the one with zero degree will be the most similar, and the one with 90 degree will be the most non-similar

We can also calculate the distance between normalized vectors as the similarity measure, i.e. the one with zero distance will be the most similar, and the one with most far apart will be the most non-similar. Moreover, when we use the distance measure, we can make use of different distance function, e.g. L1, L2.

Term – weighting Techniques

We use the tf factor – term frequency and idf factor – inverse document frequency to calculate our weighting. For tf factor, it measures the importance of an index terms within one document, i.e. if an index term appears many time in a document, then the value of tf factor of this index term will be high in this document. On other hand, idf factor is for the measurement of the importance of an index terms in all the document, i.e. if an index term appears only in a small set of document, then the idf of this index term will be high in those documents.

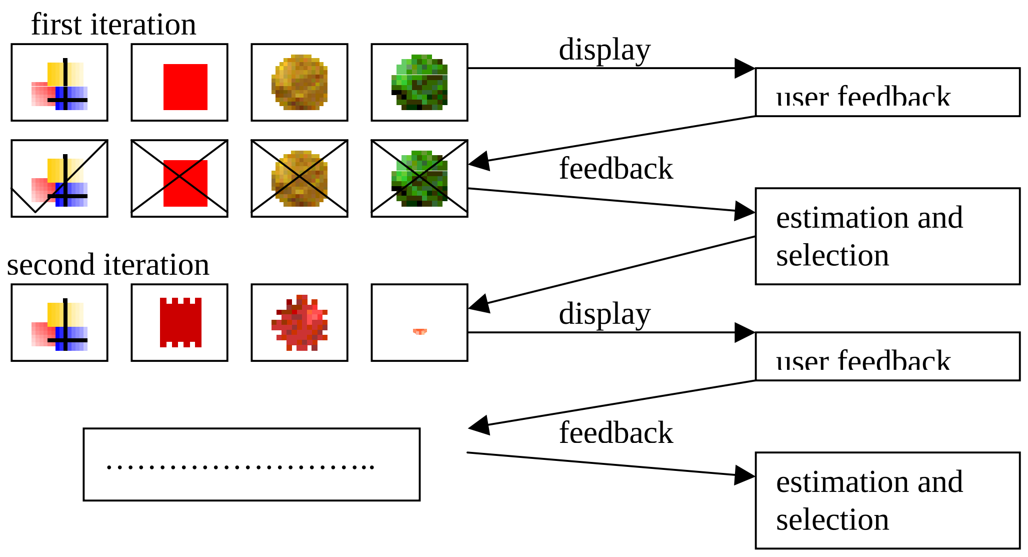
tf = (frequency of an index term in an document) / (max frequency of all index term in that document)

idf = log (number of total document) / (number of document have this index term)

9.2.3 Relevant Feedback

Relevant Feedback is used to capture user's searching behavior, and produce a better ranking next time depends on what the user selected in the search result. The reason why we need this is obvious, as our search result ranking may not totally meet the requirement of user, if there is not method to improve our search result for a certain query, then, just say an example that the first four search result for a certain query "Q" is totally no use for any user, then why we still rank them in the first four place in all the later search.

The following graph shows the overview of Relevant Feedback:



There is many ways to do Relevant Feedback, like Weight adjustment [RF 1], Bayesian method [RF 2], and Expectation maximization estimation.
As in our project, we are using Vector model, so here we just choice Weight adjustment to explain

Weight Adjustment

Main idea on weight adjustment is to change the weight of the query vector according to the set of document that user think they are relevant. Take an example; we use distance function to do our Similarity search, so we can have several methods to reform query vector: (1) Change the vector in a way that the distances of the new vector to those relevant documents' vector become closer. (2) Change the vector in a way that the distances of the new vector to those not relevant documents' vector getting fat away.

9.3 NN-search

In this part, we will discuss three different NN-search methods, A-tree, VP-tree and VA-file. NN-search is an algorithm that will speed up a search given the user specified the number of nearest point need.

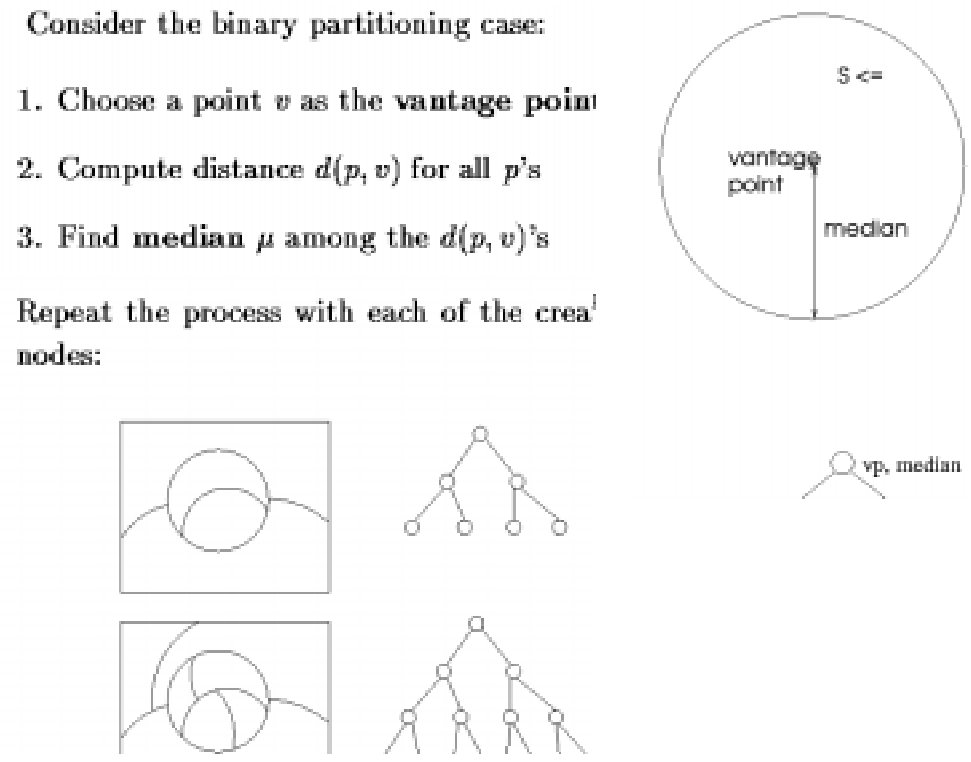
We choose this three methods for as discussion have our own reason. As usual, NN-search will involve MBR (Hyper Cube, Hyper Sphere) filtering and tree building. A-tree [NN-search 3] is an enhanced method from the R-tree [NN-search 4], which is the representative of using Hyper Cube filtering. VP-tree [NN-search 1] is the representative of using Hyper Sphere for filtering.

Moreover, VA-file [NN-search 3] is the one that do not use MBR and tree approach. In the following, we will introduce these three algorithms.

9.3.1 VP-tree (Vantage Point tree) [NN-Search 1]

VP-tree used Spheres for partition the data points, different from R-tree, which use Rectangle. As Sphere only need to store its center, so the representation of VP-tree is more compact than R-tree.

VP-tree finds vantage point for partitioning, here is the step for partitioning:

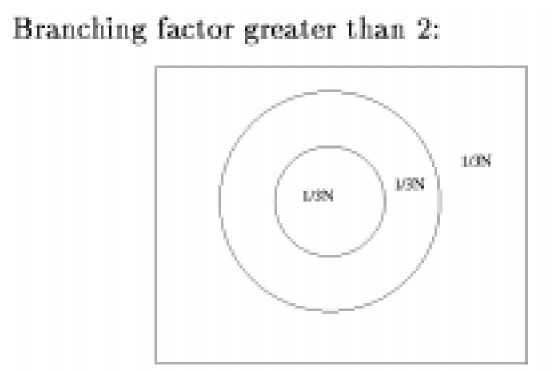


Consider the binary partitioning case:

1. Choose a point v as the **vantage poin**
2. Compute distance $d(p, v)$ for all p's
3. Find **median** μ among the $d(p, v)$'s

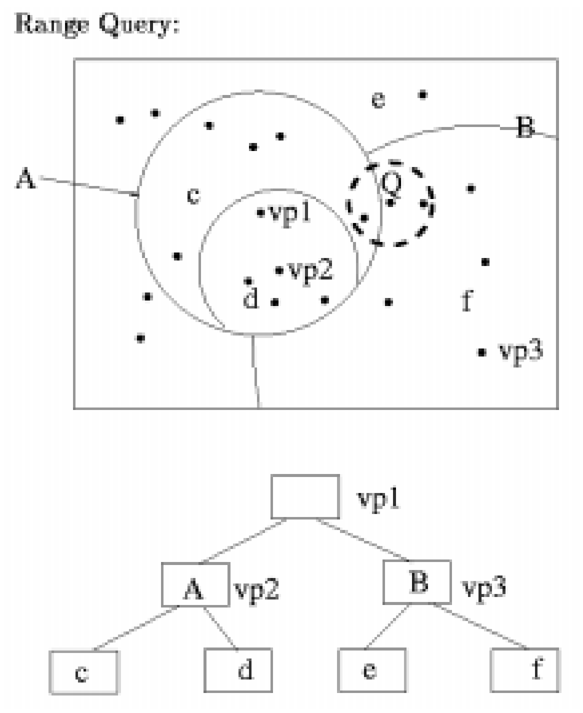
Repeat the process with each of the crea nodes:

Moreover the branching factor can be larger:



Branching factor greater than 2:

In range query, it uses the Triangle Inequality on the three distances: (1) median from vantage point (2) radius of the range query (3) distance between query point and vantage point


Range Query:
A
B
c
d
e
f
Q
vp1
vp2
vp3
vp1
A
vp2
B
vp3
c
d
e
f

9.3.2 A-tree (Approximation tree) [NN-search 2]

A-tree is an enhanced R-tree [NN-search 4] in which it introduces the idea of Virtual Bounding Rectangles (VBRs), which contain and approximate MBRs and data objects.

A-tree is using approximation in searching, but it gives exact results.

Idea of A-tree structure comes from SR-tree, in which Tree structure is used which can give a better performance in non-uniformly distribute data sets, and also filtering of children should do earlier (use VBRs to replace MBSs in SR-tree for filtering of children). Idea of VBRs is come from VA-file, in which approximation is used, so VBRs are rectangles, which approximated by their relative positions in term of its parent's MBR, so VBRs can be compactly represented.

Here is the mathematics and explain of VBRs:

$$Q_s(b_i) = a_i + \frac{(a'_i - a_i)h_s(b_i)}{q}$$

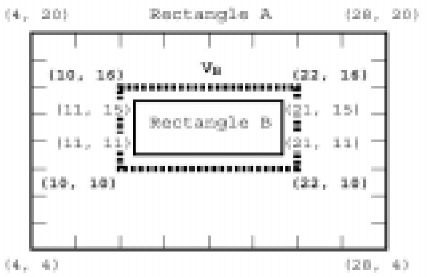
where

$$h_s(b_i) = \begin{cases} q-1 & (\text{if } b_i = a'_i) \\ \lfloor (\frac{b_i - a_i}{a'_i - a_i}) \cdot q \rfloor & (\text{otherwise}) \end{cases}$$

$$Q_e(b'_i) = a_i + \frac{(a'_i - a_i)h_e(b'_i)}{q}$$

where

$$h_e(b'_i) = \begin{cases} 1 & (\text{if } b'_i = a_i \\ \lceil (\frac{b'_i - a_i}{a'_i - a_i}) \cdot q \rceil & (\text{otherwis} \end{cases}$$



Note that $h_s(b_i) \in \{0, 1, \dots, q-1\}$, and $h_e(b'_i) \in \{1, 2, \dots, q\}$. Also, it is easy to verify that the following property holds:

$$a_i \le Q_s(b_i) \le b_i \le b'_i \le Q_e(b'_i) \le a'_i$$

Where

a_i represents the least value point of dimension "i" in the MBR

a_i' represents the greatest value point of dimension "i" in the MBR

q represents the number of partition of the MBR which specify by user

b_i represents the least value point of dimension "i" in the child MBR

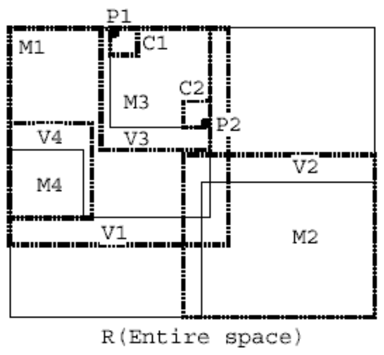
b_i' represents the greatest value point of dimension "i" in the child MBR

"h" is function for calculate the value for storing the point for the VBRs

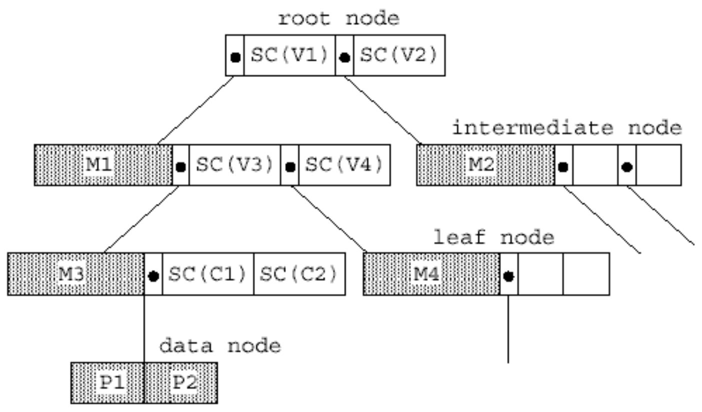
"Q" is function for reform the VBRs from the value come from "h"

Note that VBRs is just slightly large than the original MBRs and the values that come out from function "h" is at most the value of "q", so we can store it compactly.

Here is an overview structure of A-tree:



(a)VBR and MBR



(b)Tree structure

Note that each note will store the VBRs of its children. It is for filtering during searching.

9.3.3 VA-File (Vector Approximation File) [NN –search 3]

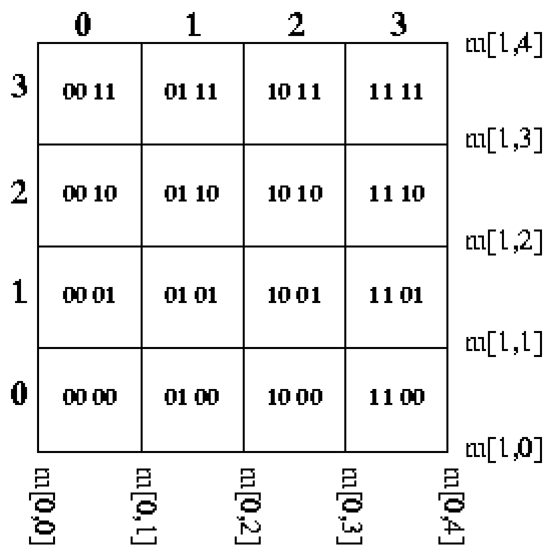
In many situations, distances to the query object are meaningless to user. so, trading result quality for reduced query execution time is completely natural. VA-File is concentrate on how to speed up the execution time of query by using approximate result instead of exact result.

Structure of VA-File

VA-File partition each dimension into 2^b intervals, in each interval contains nearly the same number of vectors.

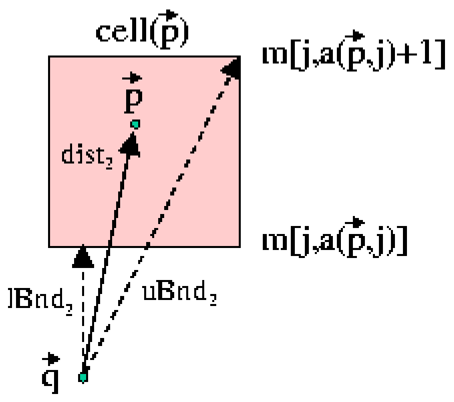
Then, each data point can be represent by an b*d bits number. For a large d, it is highly unlikely that 2 points lie in the same cell.

VA-File consists of:
i.) a vector file (store the location of intervals, e.g. m[0,0], m[0,1],….)
ii.) a file containing approximations of all vectors (store the approximate value of vectors)



Bounds on the Distance between Query and Data Point

This figure shows the lower and upper bounds of distance between a query point and a data point.



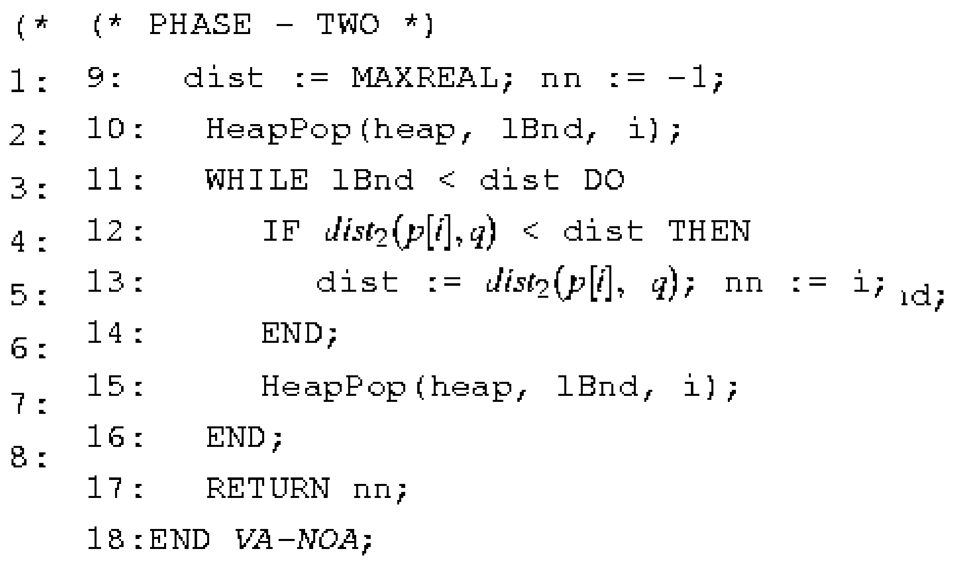
The lower and upper bounds can be calculated by using this 2 formula

$$lBnd_2(\vec{p},\vec{q}) = \sqrt{\sum_{j=0}^{d-1} \begin{cases} (m[j,a(\vec{p},j)] - q_j)^2 & q_j < m[j,a(\vec{p},j)] \\ (q_j - m[j,a(\vec{p},j)+1])^2 & q_j > m[j,a\vec{p},j)+1] \\ 0 & m[j,a(\vec{p},j)] \le q_j \le m[j,a(\vec{p},j)+1] \end{cases}}$$

$$uBnd_2(\vec{p},\vec{q}) = \sqrt{\sum_{j=0}^{d-1} \begin{cases} (m[j,a(\vec{p},j)+1] - q_j)^2 & q_j \le 1/2 \cdot (m[j,a(\vec{p},j)] + m[j,a(\vec{p},j)+1]) \\ (q_j - m[j,a(\vec{p},j)])^2 & \text{otherwise} \end{cases}}$$

Nearest Neighbor Search (NN-Search)

Use branch-and-bound technique



```
(*  (* PHASE - TWO *)
    dist := MAXREAL; nn := -1;
    HeapPop(heap, lBnd, i);
    WHILE lBnd < dist DO
        IF $dist_2(p[i], q)$ < dist THEN
            dist := $dist_2(p[i], q)$; nn := i; ıd;
        END;
        HeapPop(heap, lBnd, i);
    END;
    RETURN nn;
END $VA\text{-}NOA$;
```

Methods to reduce the execution time of searching

1. Reducing the Number of Visited Vector

    We can do this by increase the lower bound and decrease the upper bound. So that more data points will be prune out during phase one.

$$lBnd_2^l(\vec{p},\vec{q})^2 = lBnd_2(\vec{p},\vec{q})^2 + \alpha, \qquad uBnd_2^l(\vec{p},\vec{q})^2 = uBnd_2(\vec{p},\vec{q})^2 - \alpha$$

2. Avoiding Vector Visits in the Second Phase

    We can do this by completely omit the second phase and select the first k elements in the heap as the result.

Chapter 10 XML Related Technology:

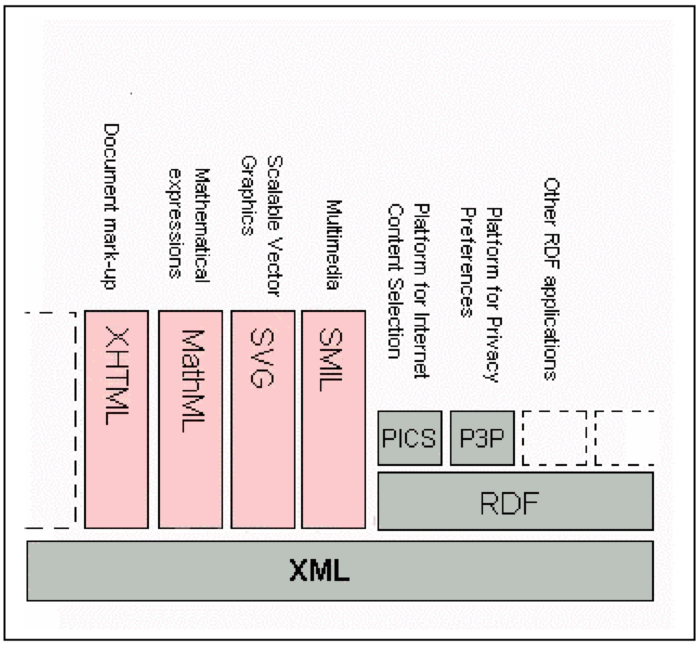
10.1 XML

Reason of Using XML in our Project:

We use XML as our data format, reason stated in following:

Main Reasons:

a. XML support a wide variety of applications. XML is beneficial to a wide variety of diverse applications: authoring, browsing, content analysis, etc. So, we can give our user a feasible and extensible way to design the data structure of their own rich information.



b. If the data is in XML format, even our user changes the structure of the data or adds new data, it will not affect our whole system process that reads or writes on the data.

c. We can use XML together with XSL and XSLT to generate XHTML, which give the user a feasible way to design the way to present their information in their own ways; on the other end, it will be more extensible on the presentation of the information, any changes will not affect the design of our whole system.

Other Reasons:

a. XML documents are easy to process, as it is machine-independent and structural. So it is easy for us to implement our system.

b. XML have been well defining in its structure, and it does not specify either semantics or a tag set; in fact XML is really a meta-language for describing markup languages. So XML can provide a facility to define tags and the structural relationships by user. As the result, XML is extendable, and the number of optional features in XML is to be kept to an absolute minimum, ideally zero.

c. XML documents are human-legible and reasonably clear. It even easy to understand the content or the meaning of an XML document without using a specific XML browser.

d. XML can be easily designed, and can be prepared and developed quickly, also creation of an XML document is easy, as it can be just edit directly in a text editor with simple shell scripts. Standards efforts are notoriously slow.

As the result, user only needs to concentrate on the design of their data and information, and at the same time, with a full control on it. Also, design, creating, modification and updating on the data and presentation method is easy and in a short time, and user don’t need to spend a large effort it. While we provide a efficient and effect way to manage their information which is in XML format, and also provide a fast way to let their client to access their richly structured documents over the LAN or WAN on PDA.

What is XML?

The Extensible Markup Language (XML) is the universal format for structured documents and data on the Web. XML in 10 points explain XML briefly [XML 2]:

1. XML is for structuring data

Structured data includes things like spreadsheets, address books, configuration parameters, financial transactions, and technical drawings. XML is a set of rules (you may also think of them as guidelines or conventions) for designing text formats that let you structure your data. XML is not a programming language, and you don't have to be a programmer to use it or learn it. XML makes it easy for a computer to generate data, read data, and ensure that the data structure is unambiguous. XML avoids common pitfalls in language design: it is extensible, platform-independent, and it supports internationalization and localization. XML is fully Unicode-compliant.

2. XML looks a bit like HTML

Like HTML, XML makes use of tags (words bracketed by '<' and '>') and attributes (of the form name="value"). While HTML specifies what each tag and attribute means, and often how the text between them will look in a browser, XML uses the tags only to delimit pieces of data, and leaves the interpretation of the data completely to the application that reads it. In other words, if you see "<p>" in an XML file, do not assume it is a paragraph. Depending on the context, it may be a price, a parameter, a person, a p... (and who says it has to be a word with a "p"?).

3. XML is text, but isn't meant to be read

Programs that produce spreadsheets, address books, and other structured data often store that data on disk, using either a binary or text format. One advantage of a text format is that it allows people, if necessary, to look at the data without the program that produced it; in a pinch, you can read a text format with your favorite text editor. Text formats also allow developers to more easily debug applications. Like HTML, XML files are text files that people shouldn't have to read, but may when the need arises. Less like HTML, the rules for XML files are strict. A forgotten tag, or an attribute without quotes makes an XML file unusable, while in HTML such practice is tolerated and is often explicitly allowed. The official XML specification forbids applications from trying to second-guess the creator of a broken XML file; if the file is broken, an application has to stop right there and report an error.

4. XML is verbose by design

Since XML is a text format and it uses tags to delimit the data, XML files are nearly always larger than comparable binary formats. That was a conscious decision by the designers of XML. The advantages of a text format are evident (see point 3), and the disadvantages can usually be compensated at a different level. Disk space is less expensive than it used to be, and compression programs like zip and gzip can compress files very well and very fast. In addition, communication protocols such as modem protocols and HTTP/1.1, the core protocol of the Web, can compress data on the fly, saving bandwidth as effectively as a binary format.

5. XML is a family of technologies

XML 1.0 is the specification that defines what "tags" and "attributes" are. Beyond XML 1.0, "the XML family" is a growing set of modules that offer useful services to accomplish important and frequently demanded tasks. Xlink describes a standard way to add hyperlinks to an XML file. XPointer and XFragments are syntaxes in development for pointing to parts of an XML document. An XPointer is a bit like a URL, but instead of pointing to documents on the Web, it points to pieces of data inside an XML file. CSS, the style sheet language, is applicable to XML as it is to HTML. XSL is the advanced language for expressing style sheets. It is based on XSLT, a transformation language used for rearranging, adding and deleting tags and attributes. The DOM is a standard set of function calls for manipulating XML (and HTML) files from a programming language. XML Schemas 1 and 2 help developers to precisely define the structures of their own XML-based formats. There are several more modules and tools available or under development.

6. XML is new, but not that new

Development of XML started in 1996 and has been a W3C Recommendation since February 1998, which may make you suspect that this is rather immature technology. In fact, the technology isn't very new. Before XML there was SGML, developed in the early '80s, an ISO standard since

1986, and widely used for large documentation projects. The development of HTML started in 1990. The designers of XML simply took the best parts of SGML, guided by the experience with HTML, and produced something that is no less powerful than SGML, and vastly more regular and simple to use. Some evolutions, however, are hard to distinguish from revolutions... And it must be said that while SGML is mostly used for technical documentation and much less for other kinds of data, with XML it is exactly the opposite.

7. XML leads HTML to XHTML

There is an important XML application that is a document format: W3C's XHTML, the successor to HTML. XHTML has many of the same elements as HTML. The syntax has been changed slightly to conform to the rules of XML. A document that is "XML-based" inherits the syntax from XML and restricts it in certain ways (e.g, XHTML allows "<p>", but not "<r>"); it also adds meaning to that syntax (XHTML says that "<p>" stands for "paragraph", and not for "price", "person", or anything else).

8. XML is modular

XML allows you to define a new document format by combining and reusing other formats. Since two formats developed independently may have elements or attributes with the same name, care must be taken when combining those formats (does "<p>" mean "paragraph" from this format or "person" from that one?). To eliminate name confusion when combining formats, XML provides a namespace mechanism. XSL and RDF are good examples of XML-based formats that use namespaces. XML Schema [XML Schema 1,2,3,4,5,6] is designed to mirror this support for modularity at the level of defining XML document structures, by making it easy to combine two schemas to produce a third which covers a merged document structure.

9. XML is the basis for RDF and the Semantic Web

XML provides an unambiguous syntax for W3C's RDF, the language for expressing metadata (in fact, for knowledge in general). RDF is like hypertext elevated to the next level. Whereas hypertext links pieces of text and leaves their relation vague, RDF can link anything and everything and assigns names to the relations: "A is the price of B" can be a relation between an object and a sum of money; "A is heavier than B" can be the relation between two sumo wrestlers; "A is the cause of B" can be the relation between a shower and your being wet. To communicate knowledge, whether in XML/RDF or in plain English, both people and machines need to agree on what words to use. A precisely defined set of words to describe a certain area of life (from "shopping" to "mathematical logic") is called an "ontology." RDF, ontologies, and the representation of meaning so that computers can help people do work are all topics of the Semantic Web Activity.

10. XML is license-free, platform-independent and well supported

By choosing XML as the basis for a project, you gain access to a large and growing community of tools (one of which may already do what you need!) and engineers experienced in the technology. Opting for XML is a bit like choosing SQL for databases: you still have to build your own database and your own programs and procedures that manipulate it, and there are many tools available and many people who can help you. And since XML is license-free, you can build your own software around it without paying anybody anything. The large and growing support means that you are also not tied to a single vendor. XML isn't always the best solution, but it is always worth considering.

The design goals for XML [XML 1] **are:**

a. XML shall be straightforwardly usable over the Internet.

b. XML shall support a wide variety of applications.

c. XML shall be compatible with SGML.

d. It shall be easy to write programs which process XML documents.

e. The number of optional features in XML is to be kept to the absolute minimum, ideally zero.

f. XML documents should be human-legible and reasonably clear.

g. The XML design should be prepared quickly.

h. The design of XML shall be formal and concise.

i. XML documents shall be easy to create.

j. Terseness in XML markup is of minimal importance.

XML will [XML 4]:

a. Enable internationalized media-independent electronic publishing.

b. Allow industries to define platform-independent protocols for the exchange of data, especially the data of electronic commerce.

c. Deliver information to user agents in a form that allows automatic processing after receipt.

d. Make it easier to develop software to handle specialized information distributed over the Web.

e. Make it easy for people to process data using inexpensive software.

f. Allow people to display information the way they want it, under style sheet control.

g. Make it easier to provide metadata -- data about information -- that will help people find information and help information producers and consumers find each other.

Logical Structures of XML [XML 1]:

Following will give a brief description on the structure of XML.

Element

[Definition: Each XML document contains one or more elements, the boundaries of which are either delimited by start-tags and end-tags, or, for empty elements, by an empty-element tag. Each element has a type, identified by name, sometimes called its "generic identifier" (GI), and may have a set of attribute specifications.] Each attribute specification has a name and a value.

```
element ::=   EmptyElemTag
              | STag content ETag       [WFC: Element Type Match]
                                        [VC: Element Valid]
```

This specification does not constrain the semantics, use, or (beyond syntax) names of the element types and attributes, except that names beginning with a match to (('X'|'x')('M'|'m')('L'|'l')) are reserved for standardization in this or future versions of this specification.

Well-formalness constraint (WFC): Element Type Match

The Name in an element's end-tag must match the element type in the start-tag.

Validity constraint (VC): Element Valid

a. An element is valid if there is a declaration matching elementdecl where the Name matches the element type, and one of the following holds:

b. The declaration matches EMPTY and the element has no content.

c. The declaration matches children and the sequence of child elements belongs to the language generated by the regular expression in the content model, with optional white space (characters matching the nonterminal S) between the start-tag and the first child element, between child elements, or between the last child element and the end-tag. Note that a CDATA section containing only white space does not match the nonterminal S, and hence cannot appear in these positions.

d. The declaration matches Mixed and the content consists of character data and child elements whose types match names in the content model.

e. The declaration matches ANY, and the types of any child elements have been declared.

White Space

S (white space) consists of one or more space (#x20) characters, carriage returns, line feeds, or tabs.

```
S       ::=       (#x20 | #x9 | #xD | #xA)+
```

Start-Tags

Definition: The beginning of every non-empty XML element is marked by a start-tag.]

```
Start-tag
    STag          ::=     '<' Name (S Attribute)* S? '>' [WFC: Unique Att
    Spec]
    Attribute     ::=     Name Eq AttValue [VC: Attribute Value Type]
      [WFC: No External Entity References]
      [WFC: No < in Attribute Values]
```

The Name in the start- and end-tags gives the element's type. [Definition: The Name-AttValue pairs are referred to as the attribute specifications of the element], [Definition: with the Name in each pair referred to as the attribute name] and [Definition: the content of the AttValue (the text between the ' or " delimiters) as the attribute value.]Note that the order of attribute specifications in a start-tag or empty-element tag is not significant.

Well-formedness constraint (WFC): Unique Att Spec

No attribute name may appear more than once in the same start-tag or empty-element tag.

Validity constraint (VC): Attribute Value Type

The attribute must have been declared; the value must be of the type declared for it.

Well-formedness constraint (WFC): No External Entity References

Attribute values cannot contain direct or indirect entity references to external entities.

Well-formedness constraint (WFC): No < in Attribute Values

The replacement text of any entity referred to directly or indirectly in an attribute value must not contain a <.

An example of a start-tag:

<termdef id="dt-dog" term="dog">

[Definition: The end of every element that begins with a start-tag must be marked by an end-tag containing a name that echoes the element's type as given in the start-tag:]

End-tag

ETag ::= '</' Name S? '>'

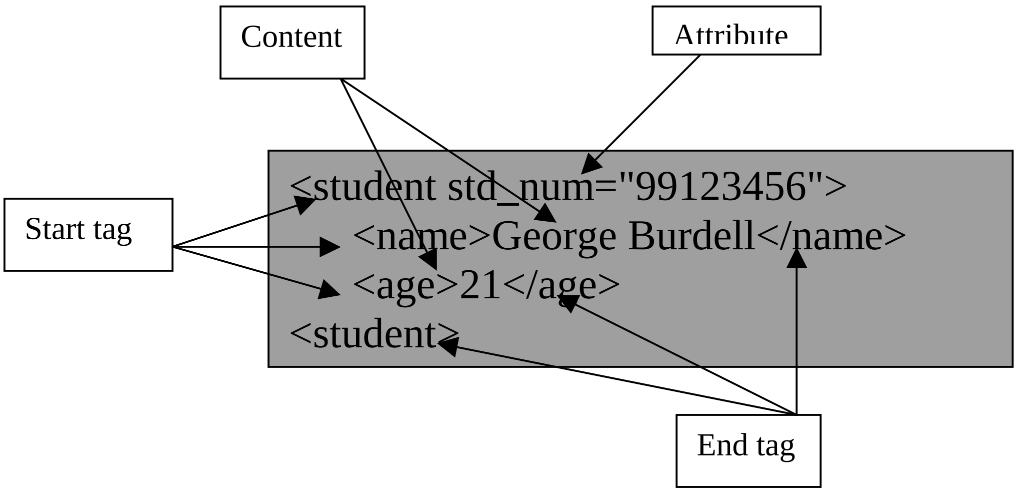
An example of an end-tag:

</termdef>

Content of Elements

[Definition: The text between the start-tag and end-tag is called the element's content:]
content ::= CharData? ((element | Reference | CDSect | PI | Comment) CharData?)* /* */

Overall Example:



10.2 XSL and XSLT:

Reason of Using XSL and XSLT in our Project:

We use XSL and XSLT as method for translation of information, reason stated in following:

Main Reasons:

a. XSL support a wide variety of applications. It provides a standard and powerful way to transform from XML format data to another informative Formal. So, we can give our user a feasible and extensible way to design the presentation of their own data.

b. Even our user wants to have changes or have new way to present their data, it will not affect our whole system processing on it, so it is probably easier the to have maintenance.

c. As XSL can give user a different transformation of their XML at different node level, so that the transformation results of a XML document can be different depending on how specify of the information that user's client needs.

We really don't want to limit the design of the presentation of user's rich data, so we need to provide a feasible, standardized, and extensible solution to them, here we choose XML together with XSL.

What is XSL and XSLT [XSL 3]**?**

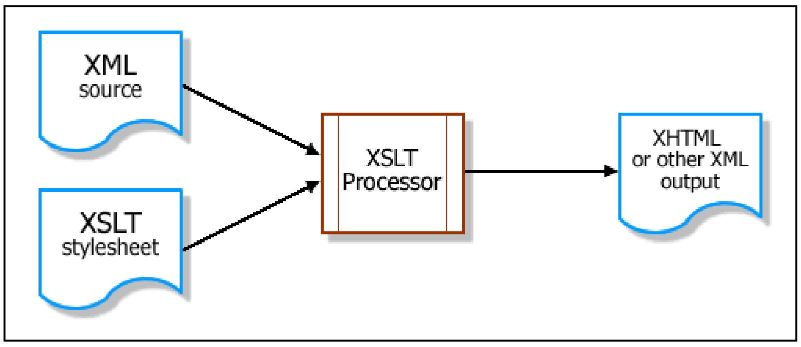
XSL is a language for expressing style sheets. An XSL style sheet is, like with CSS, a file that describes how to display an XML document of a given type. XSL shares the functionality and is compatible with CSS2 [CSS2](although it uses a different syntax). It also adds:

A transformation language for XML documents: XSLT. Originally intended to perform complex styling operations, like the generation of tables of contents and indexes, it is now used as a general purpose XML processing language. XSLT is thus widely used for purposes other than XSL, like generating HTML web pages from XML data.

Advanced styling features, expressed by an XML document type which defines a set of elements called Formatting Objects, and attributes (in part borrowed from CSS2 properties and adding more complex ones. See also the XSL-FO page.

How Does It Work [XSL 3]**?**

Styling requires a source XML documents, containing the information that the style sheet will display and the style sheet itself which describes how to display a document of a given type.



Multiple XSLT style sheets can present a single document in multiple formats. A single style sheet could transform multiple instances of one data type into a standard presentation format, which you could change simply by modifying the style sheet. Or XSLT could transform multiple instances of data into multiple formats. And it is not constrained to presentation: XSLT is a powerful tool for translating one system's data format into another's.

The following shows a sample process of how XSL works, that means how XML file and how it can be transformed and rendered. [XSL 4]

a. Get the XSLT processor:

To get started, we need an XSLT processor. There are only a handful of desktop XSLT prototyping tools available, as the majority of such tools are for full-scale production systems.

Recent browsers such as Internet Explorer 5.5 & 6.0, Netscape 6.1, and Mozilla support XSLT processing. They are probably the easiest tools to use but are currently unreliable in their specification support. Also, a browser does not provide the support of a true development tool and won't help much when debugging code. So, in our system, XSLT conversions are done on the server, so browsers will work only for browser the generated HTML.

As we find, Instant Saxon is a simple, command-line server-style XSLT processor for Windows, it provides basic file output and error information and it delivers more solid XSLT support than is available from browsers. Although it do not support a full-blown development environment, Instant Saxon is a great tool on it.

b. Example:

order.xml:

```
<?xml version="1.0" encoding="utf-8"?>
<?xml-stylesheet type="text/xsl" href="./order.xsl"?>

<order number="734" name="Mike">
   <line-item>
         <description>Cheeseburger</description>
         <unit-price>1.59</unit-price>
         <quantity>2</quantity>
   </line-item>
   <line-item>
         <description>Large french fries</description>
         <unit-price>1.09</unit-price>
         <quantity>1</quantity>
   </line-item>
   <line-item>
         <description>Medium Coke</description>
         <unit-price>.99</unit-price>
         <quantity>1</quantity>
   </line-item>
</order>
```

order.xsl:

```
<?xml version="1.0" encoding="utf-8"?>
<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
version="1.0">
<xsl:output method="xml" indent="yes" omit-xml-declaration="yes"/>

<xsl:template match="order">

<html>
<head><title>Your order</title></head>
<body>

<xsl:value-of select="@name"/>’s Order (#<b><xsl:value-of
select="@number"/></b>): <br/><br/>

<table border="1">
<tr>
   <td><b>Item</b></td>
   <td><b>Amount</b></td>
   <td><b>Cost</b></td>
   <td><b>Total</b></td>
</tr>

<xsl:for-each select="line-item">
```

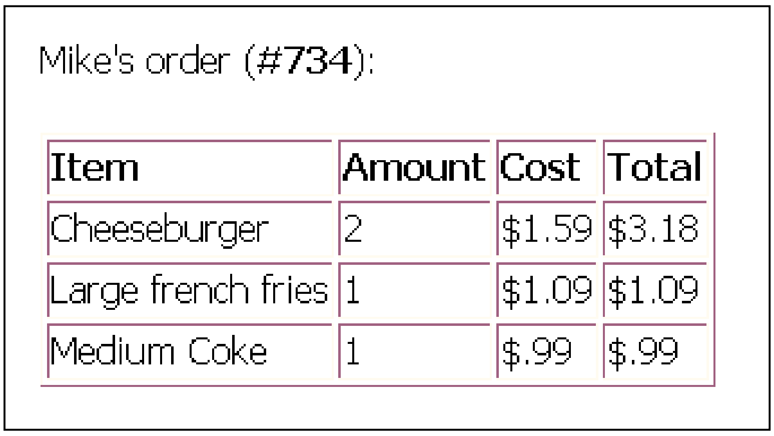
```
<tr>
  <td><xsl:value-of select="description"/></td>
  <td><xsl:value-of select="quantity"/></td>
  <td>$<xsl:value-of select="unit-price"/></td>
  <td>$<xsl:value-of select="quantity * unit-price"/></td>
</tr>
</xsl:for-each>

</table>
</body>
</html>

</xsl:template>
</xsl:stylesheet>
```

The XML file is linked to the XSL style sheet, so we can view the XML file in a suitable browser or XSL Transform it in XML Spy. With Instant Saxon, we can execute a command, and translate the XML document into an HTML document:

```
saxon.exe order.xml order.xsl > order.html
```

This will write the transformed HTML output into a file called order.html that we can view in our browser.

The results of the example looks like this:



Mike's order (#734):

Item	Amount	Cost	Total
Cheeseburger	2	$1.59	$3.18
Large french fries	1	$1.09	$1.09
Medium Coke	1	$.99	$.99

An HTML page with a title showing Mike's order (number 734) and a table of what he ordered, including cost. The XSLT processor took the XML file containing the data and transformed it into the HTML output. The XSLT style sheet defined the HTML tags to place around the XML data using the processing instructions that comprise the XSLT language.

We can also make an XML document have its own default XSLT style sheet rather than normally instructs which style sheet to apply by including the line

```
<?xml-stylesheet type="text/xsl" href="my.xsl"?>
```

where my.xsl is a URL to the style sheet. This code is essential for browser-based transforming.

c. Sample explanation:

The main ideas of XSLT are build up a context--which is a particular node or set of nodes in an XML document, and output a formatted version of the data that exists within that context. To do this, an XSLT style sheet is separated into discrete templates, each of which handles certain types of tags in the XML document. Within these templates, XSLT utilizes variables, passed parameters, looping constructs, conditionals, and other devices geared toward transforming XML.

The <xsl:stylesheet> element is the outermost element of any XSLT style sheet, assigning it a version and one or more namespaces:

```
<xsl:stylesheet version="1.0"
xmlns:xsl="http://www.w3.org/1999/XSL/Transform>
...
</xsl:stylesheet>
```

You can set other attributes, but for almost all basic style sheets, you can use these exact <xsl:stylesheet> tags. The template elements are nested within them.

d. XSLT templates

The <xsl:template> element defines a context in which to execute along with the resulting output. It has the syntax

```
<xsl:template
  match="expression"
  name="name"
  priority="number"
  mode="mode">
</xsl:template>
```

An XSLT processor executes a <xsl:template> when it finds either an explicit call in the style sheet or a matching node in the source XML document. The most common cause is matching nodes encountered as the XSLT processor scans the XML. The match attribute takes an XPath expression [XPATH 1], identifying which nodes set off the template.

We can have more than one template matches a node. In this case, fairly complex rules need, e.g. using the mode and priority attributes determine which template will process the node. So a simple style sheets contain only one template to match a given node is better.

There are two usual cases for XSLT, (1) for XML documents that contain mostly marked-up text, such as HTML, XSLT style sheet will contain a template for each tag we might encounter. (2) For XML documents that contain highly structured hierarchical data, our style sheet need to contain templates for only the top-level nodes. These templates will know the data structure and access the subnodes directly, rather than leave them to other templates.

For example, the following sample XML file:

```
<?xml version="1.0" encoding="UTF-8"?>
<?xml-stylesheet type="text/xsl" href="./sample.xsl"?>
<book>
   <title>Stuff Happens</title>
   <chapter type="prologue">
          <title>How it begins</title>
          <paragraph>At first there’s nothing going on.</paragraph>
          <paragraph>Eventually that all changes.</paragraph>
   </chapter>
   <chapter>
          <title>What transpires</title>
          <paragraph>Things occur, one after the other.</paragraph>
          <paragraph>Some are simultaneous or overlap.</paragraph>
   </chapter>
   <chapter type="epilogue">
          <title>Where it ends</title>
          <paragraph>Everything comes to a stop.</paragraph>
          <paragraph>Or at least nothing else happens.</paragraph>
   </chapter>
</book>
```

contains a short, marked-up book. It consists of one <book> node containing a <title> and multiple <chapter> nodes. This template would execute for each <chapter> within the top-level <book>:

```
<xsl:template match="/book/chapter">
This is chapter <xsl:number/>, entitled "<xsl:value-of select="title"/>"
</xsl:template>
```

If an XSLT processor do not match any template for a node or its parent nodes, it simply outputs the node’s contents, even these can contain subnodes that they have their own templates. So a style sheet with only the preceding template would produce the following result:

```
<?xml version="1.0" encoding="utf-8"?>
Stuff Happens
This is chapter 1, entitled "How it begins"
This is chapter 2, entitled "What transpires"
This is chapter 3, entitled "Where it ends"
```

The <paragraph> nodes are skipped because their <chapter> parents were processed, but the first <title>, not being in a <chapter>, just gets printed as is.

e. XPath expressions

XPath have an important role in XSL. XPath is a language for referencing specific parts of an XML document, although it supports a richer feature set than just simply pointing to data. In XSLT style sheets, XPath expressions return four types of values: node-set, Boolean, number, and string. XSLT elements take an XPath expression as an attribute value in order to get the result of the evaluated expression.

The most common use in basic XSLT is to return a node-set or string, depending on the element. For example, <xsl:template match="chapter"> defines a template for <chapter> nodes within the current node context. In this case, the XPath expression chapter returns a node-set as the new context for further XSL functions. Whereas in <xsl:value-of select="title"/>, the XPath expression title returns the raw content of any <title> nodes within the current context as a string.

f. Others

Just like other programming language, XSL also can:
(1) declaring parameters and variables (by xsl:param / xsl:with-param, xsl:variable),
(2) calculation of value (by xsl:value-of, xsl:number),
(3) Looping and sorting (by xsl:for-each, xsl:sort),
(4) Having Conditional statements (by xsl:if, xsl:choose / xsl:when / xsl:otherwise)

More detail examples and explanations can be found at [XSL 1], [XSL 2], [XSL 3], [XSL 4], [XSLT 1], [XPATH 1]

10.3 XHTML and XHTML base:

Reason of Using XHTML and XHTML base in our Project:

We use XHTML and XHTML base as method for presenting information, reason stated in following:

Main Reasons:

a. Both of them can be the product after XSLT, and can be just browse on nowadays browsers.
b. Parsing of them will be much more efficient than man-made HTML, as the later usually make mistake on the rule of HTML, it makes the browser work smoother.

c. Good for future development of the HTML-based browsers, as new features can be standardize by the XSLT

What is XHTML [XHTML 1] **?**

XHTML is a reformulation of HTML 4 [HTML 1] as an XML 1.0 application, and three DTDs [XML DTD 1] corresponding to the ones defined by HTML 4. The semantics of the elements and their attributes are defined in the W3C Recommendation for HTML 4. These semantics provide the foundation for future extensibility of XHTML

XHTML is a family of current and future document types and modules that reproduce, subset, and extend HTML 4. XHTML family document types are XML based, and ultimately are designed to work in conjunction with XML-based user agents. The details of this family and its evolution are discussed in more detail in the section on Future Directions.

Developers who migrate their content to XHTML 1.0 will realize the following benefits:

a. XHTML documents are XML conforming. As such, they are readily viewed, edited, and validated with standard XML tools.
b. XHTML documents can be written to operate as well or better than they did before in existing HTML 4-conforming user agents as well as in new, XHTML 1.0 conforming user agents.
c. XHTML documents can utilize applications (e.g. scripts and applets) that rely upon either the HTML Document Object Model or the XML Document Object Model [DOM].
d. As the XHTML family evolves, documents conforming to XHTML 1.0 will be more likely to interoperate within and among various XHTML environments.

The XHTML family is the next step in the evolution of the Internet. By migrating to XHTML today, content developers can enter the XML world with all of its attendant benefits, while still remaining confident in their content's backward and future compatibility.

Reasons to write XHTML today [XHTML 3]**:**

a. XHTML 1.0's design is intended to curry good habits among Web page authors. Many of us have taken advantage of the quirks in browser behavior to produce a specific visual result when using nonstandard HTML. The problem with this approach is: First, the result depends on the visitor using the same browser as the designer. This method also let to non-termination of demand for a browser to anticipate and render the author's intention, despite the nonstandard use of HTML.

   For example, it is common to find incorrectly nested elements in Web pages produced by authors working in both text editors and some authoring tools. You may see:

<p>I'm going to <b>nest<i>bold and italics</b></i>elements.

Web browsers have long understood this sample to mean that the author wanted to give both bold and italics treatment to the phrase bold and italics, despite the fact that the <b> element is erroneously closed before the <i> element. Many designers would say that since the sample will work in their intended browsers, they don't need to make any corrections. However, in more substantive examples, even the two most popular browsers don't behave in the same manner. Consider the case of a missing </table> tag at the end of a table. Internet Explorer "helpfully" infers the author's intention when it encounters the next block element after the point where the table should have been closed and displays the table as if the closing tag were present. Netscape Navigator, on the other hand, hasn't been programmed to handle such an error and instead refuses to display any of the table content at all, something quite disconcerting when reported to authors who view their work only in IE before publishing.

b. Today's commercial Web sites are far more likely to be using dynamic page-generation systems than ever before. Major content sections are retrieved from databases or document storage and delivery systems, and advertising may then be brought in from a third-party site. Each of these pieces needs a solid and dependable place to bind to the template for delivery. A sloppy, miscoded page will complicate matters, not only for proper insertion of dynamic content, but also in the rendering of the page for the end user.

c. Web pages that conform with the XHTML Recommendation display faster because the browser has no guesswork to do. It simply parses the document according to the rules of XHTML and finds no place where the author's intent is unclear. Especially for the device with less powerful CPU likes PDA, the waiting time for loading a page will be much shorter.

XHTML Base, a subset of HTML [XHTML 2]**:**

The XHTML Basic document type includes the minimal set of modules required to be an XHTML host language document type, and in addition it includes images, forms, basic tables, and object support. It is designed for Web clients that do not support the full set of XHTML features; for example, Web clients such as mobile phones, PDAs, pagers, and settop boxes. The document type is rich enough for content authoring.

XHTML Basic is designed as a common base that may be extended. For example, an event module that is more generic than the traditional HTML 4 event system could be added or it could be extended by additional modules from XHTML Modularization such as the Scripting Module. The goal of XHTML Basic is to serve as a common language supported by various kinds of user agents.

Information appliances are targeted for particular uses. They support the features they need for the functions they are designed to fulfill. The following are examples of different information appliances:

a. Mobile phones
b. Televisions
c. PDAs
d. Vending machines
e. Pagers
f. Car navigation systems
g. Mobile game machines
h. Digital book readers
i. Smart watches

Chapter 11 Review on Trial Application

11.1 Building our First Application at Pocket PC:

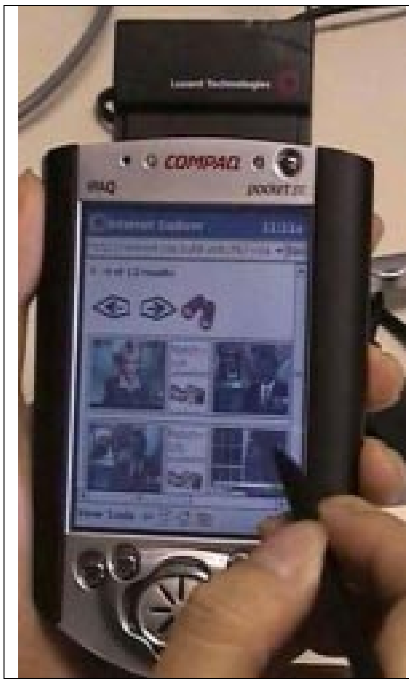
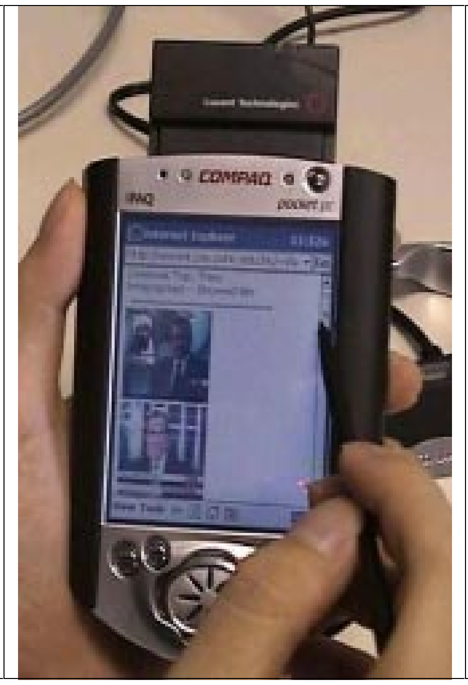
Introduction:

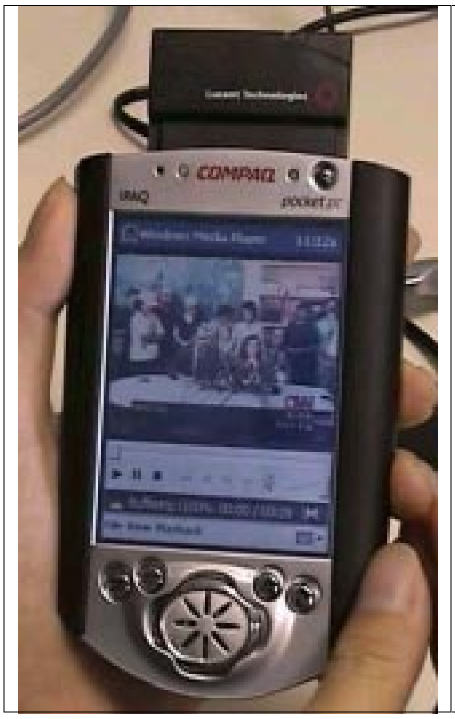
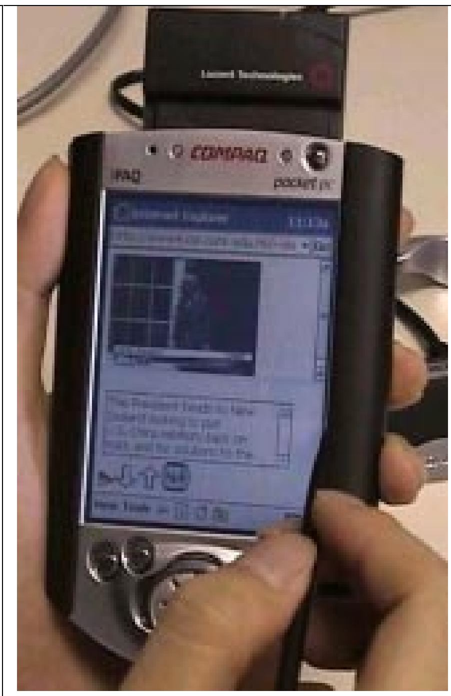
Our first application is a Video Digital Video Library on PDA, which supports:

Searching of key word in news report
Display of the main frame as the search result
Have a abstracts and key frame of each news report
Have a news report reading mode
Have Steaming of Video of news report

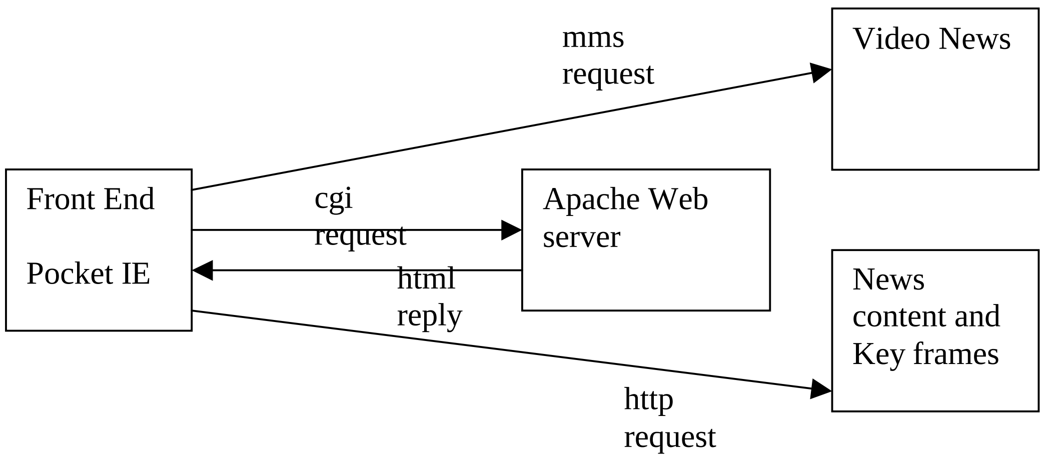
Pictures of the First Application at Pocket PC:

COMPAQ iPAQ pocket pc Internet Explorer Search Reset	Search page – searching of key word in news report

	Result page – Display of the main frame as the search result		Abstract – abstracts and key frame of each news report

	Video Steaming of news report		News report reading mode

11.2 Architecture of First Application at Pocket PC



Pocket IE of Pocket PC:

Due to the powerful functions that already exists in Internet Explore of Microsoft Windows at Desktop, we then study on it. We found that it miss two thing that important to our application (1) Support of Jscript: although Pocket PC claim that it supports Jscript , but it was just a very limit support, as the part of DHTML are totally not supported (2) No ActiveX control: this make the video need to be play outside the Pocket IE, and we can’t control it through out the Pocket IE

11.3 Bluetooth and 3G technology:

As our Application needs to have a wireless support, so we found that both Bluetooth and 3G technology will solve the problem of high power consumption of Wireless LAN card. So we also have a study and test on both of it.

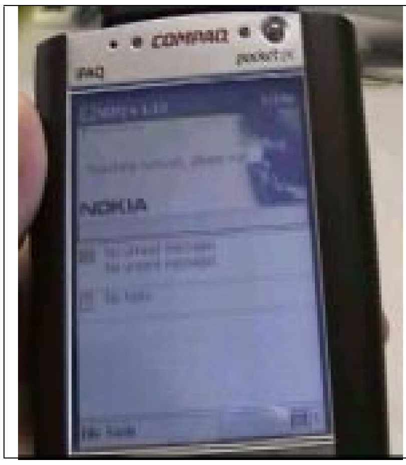
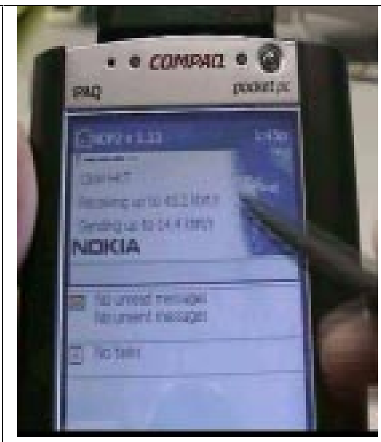
Here is a brief summery on it:

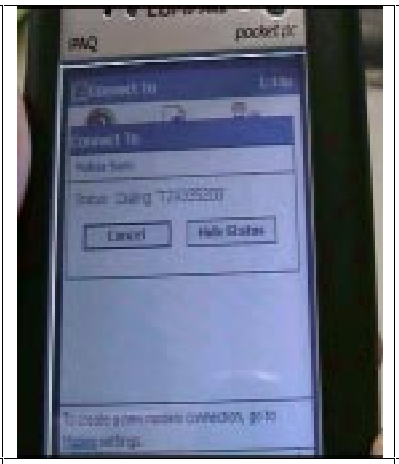
2G	GSM data network 14.4kbps GSM High-Speed Circuit Switching Data (HSCSD) 43.2kbps CDMA data network 64kbps
2.5G	General Packet Radio Service (GPRS) 40kbps
3 G	Third Generation wireless where high-speed, broadband mobility, about 64kbps
Bluetooth	low cost, low power consumption ,short distance, about 1 Mbps

Picture on trying of both Technologies:

GSM:

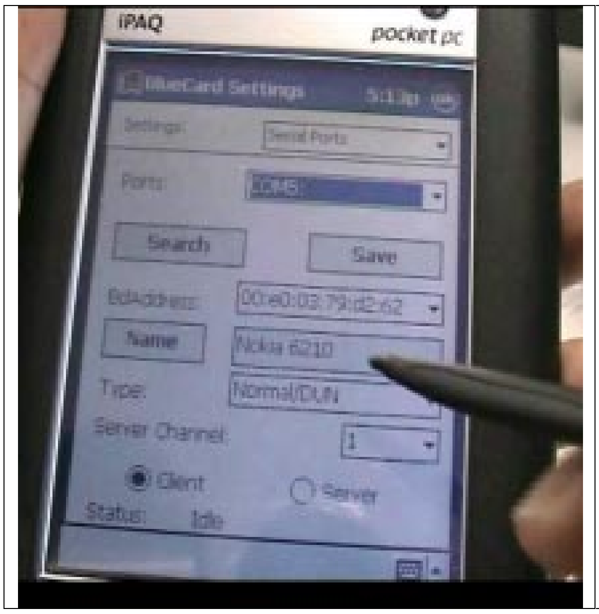
	GSM phone card		Using GSM phone card for wireless communication

	Make communication between the Pocket PC and the GSM card phone	 COMPAQ pocket pc iPAQ NOKIA No unread messages No urgent messages No tasks	Service available

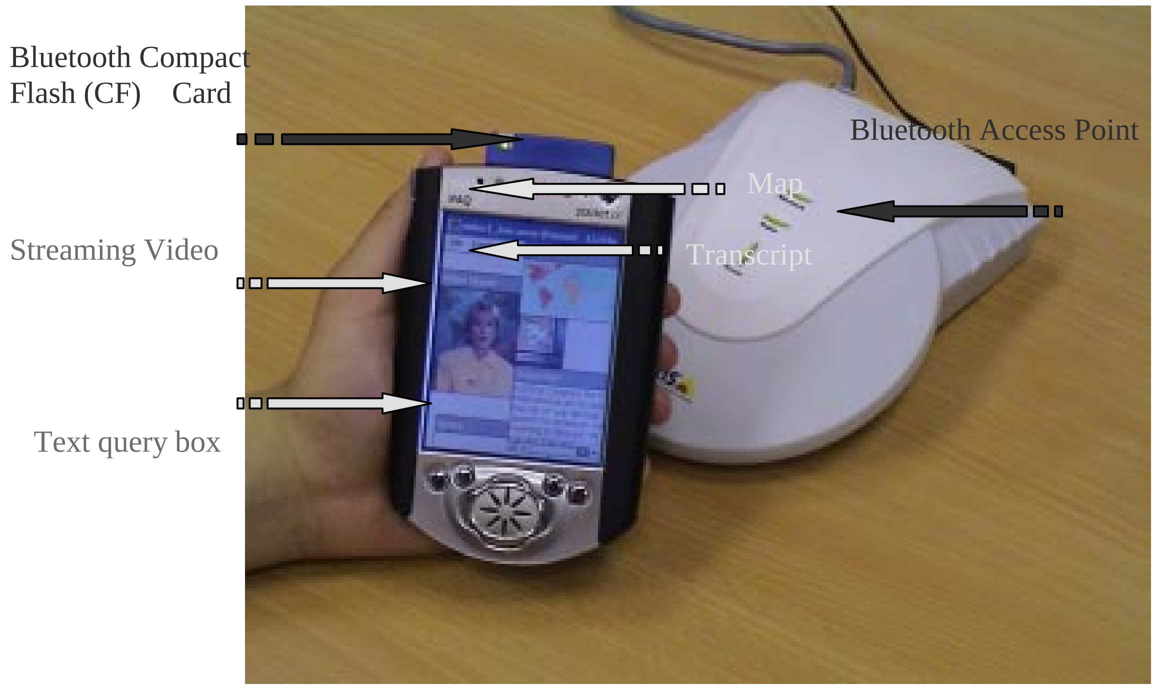
	Prepare to have a dial up connection to internet		Dialing up to the ISP
	Trying the internet service with 2G support		

Bluetooth mobile phone:

	NOKIA 6120 Mobile phone which support Bluetooth communication		Pocket PC which also have support on Bluetooth

 iPAQ pocket pc BlueCard Settings 5:13p Settings: Serial Ports Ports: [illegible] Search Save BdAddress: 00:e0:03:79:d2:62 Name: Nokia 6210 Type: Normal/DUN Server Channel: 1 ● Client ○ Server Status: Idle	Two Bluetooth card establish communication with each other, we can then use dial up service of the NOKIA 6120 in order to have Internet service

Bluetooth access point:



11.4 Problem in our First Application

a. It is not extensible. The overall application cannot be more than what we have implemented, and a set of CGI program which is difficult to add new things in it. Also our application’s functionality is limit by the power of Pocket IE.

b. Playing of video need to jump out our Application, and we can’t control on the playing of it

c. Searching of key words in the Web site is just use Sequential search in the files; searching time will be increase as the size of the information growth.

d. Changing in format of display for the information needs to have an overall change of Server part. Maintenance on the Application become difficult.

These experiences give us the direction to design our “real” FYP project

Chapter 12 Conclusion and Personal Contribution on Project

12.1 Major responsibility on the project

1. General Design

   After the trial application, I and my partner have the general idea of making a generic server for providing information to PDA user. After we found out there is HTML viewer supported by the Pocket PC. We have design to using XML data as input data format and using XSL to format the output of the data.

2. Client Side programming

   On the Implementation phase, my major responsibility is to implement the client program in the Pocket PC. The most different part is to learn the APIs for the Window CE with limited reference resource.

3. Define communication format between server and client

   When design the communication format between server and client, we choose XML as data format between them, using XML format because it is a plain text format and so allow the interoperability for building client on other platform.

12. 2 TimeLine

June ~ Aug 2001

1. Study the different between two PDA OS

   We have try to use two kind of PDA OS, try to find out what functions are they provide. We also try to search on the web about the comparison of the two different PDA OS. We find out the palm OS is more popular in use.

2. Try Palm OS programming

   To get more understanding of the PDA OS, we use a few weeks to learn programming on Palm OS. We find that there is a large limitation on Palm OS, there is not enough support of multimedia content and not easy to build multi-window program.

3. Try Pocket PC programming

After trying the Palm OS, we try to programming on Windows CE. Pocket PC PDA provides more build-in libraries supporting multimedia content and also the programming environment is similar desktop PC.

4. Build the trial application

In the late summer, we use the Pocket IE (Pocket Internet Explorer) to build a trial program. It is only a JavaScript base client with simple CGI server. Build this application is try to get more experience on providing information to user.

Sep ~ Dec 2001

1. Design this application

At the start of the first semester, we get an idea of building a generic server for providing information to PDA. We design there are three windows provide information to user, and what functions are they provide.

2. Design the system architecture

After we design our application, Wireless Digital Information System, we design the architecture of our system. To support different data structure for a generic server, we have design to using XML data as input data format and using XSL to format the output of the data. Also we design to use database to store the XML data.

3. Build the Pocket PC client

At the same time, I start to build the PDA client. I try to implement the control component, the main window, and movie window. The most important difficulties are lack of reference about the library and the setting of programming tools.

Jan ~ Apr 2002

1. Define communication format between server and client

In the second semester, we have defined the communication between server and client. XML was chosen as data format between them, using XML format because it is a plain text format and so allow the interoperability for building client on other platform.

2. Finish the Pocket PC client

In this semester, I finished the implement the implement the function of image window. In the other hand, I need to handle the XML format communication between the client and server, as the XML parser is not well supported by the Window CE, I just parser XML with my own code.

3. Data collection and preparation

   We need to generate a set of XML data as input of server for testing and demonstration. We use information about our department to generate the XML data set.

12.3 Conclusion on the project

In this project, we have built a generic system for providing information to PDA Client with search function. During the process, we have learn a series knowledge and techniques including PDA programming, XML handling, searching engine.

Chapter 13 Reference

[XML 1]
"Extensible Markup Language (XML) 1.0 (Second Edition)"
W3C Recommendation 6 October 2000
Available at http://www.w3.org/TR/2000/REC-xml-20001006

[XML 2]
"XML in 10 points"
Bert Bos
Revised 13 Nov. 2001 (last update: $Date: 2001/11/21 19:45:05 $)
Created 27 Mar 1999
Available at http://www.w3.org/XML/1999/XML-in-10-points.html

[XML 3]
"Namespaces in XML"
World Wide Web Consortium 14-January-1999
Available at http://www.w3.org/TR/1999/REC-xml-names-19990114/

[XML 4]
"Extensible Markup Language (XML) Activity Statement"
1996-2001 W3C® (MIT, INRIA, Keio)
Available at http://www.w3.org/XML/Activity

[XSL 1]
"The Extensible Stylesheet Language (XSL)"
Max Froumentin, Team Contact for the XSL Working Group
$Id: 2001/07/10 09:53:29 $
Available at http://www.w3.org/Style/XSL/

[XSL 2]
"Extensible Stylesheet Language (XSL) Version 1.0"
W3C Recommendation 15 October 2001
Available at http://www.w3.org/TR/xsl

[XSL 3]
"What is XSL?"
Available at http://www.w3.org/Style/XSL/WhatIsXSL.html

[XSL 4]
"Introduction to XSLT"
Available at http://builder.cnet.com/webbuilding/0-3882-8-7033236-1.html

[XSL 5]
"XSL Transformations"
An excerpt from New Riders Inside XML by Steven Holzner.
Available at http://www.vbxml.com/xsl/articles/xsl_transformations/default12.asp

[XSLT 1]
"XSL Transformations (XSLT) Version 1.0"

W3C Recommendation 16 November 1999
Available at http://www.w3.org/TR/xslt

[XPATH 1]
"XML Path Language (XPath) Version 1.0"
W3C Recommendation 16 November 1999
Available at http://www.w3.org/TR/xpath

[XQuery 1]
"XML Query"
Massimo Marchiori
Created April 2000
$Revision: 1.56 $ $Date: 2001/10/23 08:47:37 $
Available at http://www.w3.org/XML/Query

[XQuery 2]
"XQuery 1.0: An XML Query Language"
W3C Working Draft 07 June 2001
Available at http://www.w3.org/TR/xquery/

[CSS2 1]
"Cascading Style Sheets, level 2 CSS2 Specification"
W3C Recommendation 12-May-1998
Available at http://www.w3.org/TR/REC-CSS2/cover.html

[XHTML 1]
"XHTML™ 1.0: The Extensible HyperText Markup Language"
A Reformulation of HTML 4 in XML 1.0
W3C Recommendation 26 January 2000
Availabe at http://www.w3.org/TR/xhtml1/

[XHTML 2]
"XHTML™ Basic"
W3C Recommendation 19 December 2000
http://www.w3.org/TR/xhtml-basic/

[XHTML 3]
"XHTML: Past, present, and future "
By Ann Navarro (9/12/01)
http://builder.cnet.com/webbuilding/0-3881-8-7080997-1.html?tag=st.bl.3881.edt.3881--7080997-1

[XML Schema 1]
"XML Schema Part 0: Primer "
W3C Recommendation, 2 May 2001
http://www.w3.org/TR/xmlschema-0/

[XML Schema 2]
“XML Schema Part 1: Structures”
W3C Recommendation 2 May 2001
http://www.w3.org/TR/xmlschema-1/

[XML Schema 3]
“XML Schema Part 2: Datatypes”
W3C Recommendation 02 May 2001
http://www.w3.org/TR/xmlschema-2/

[XML Schema 4]
“XML Schema Requirements”
W3C Note 15 February 1999
http://www.w3.org/TR/NOTE-xml-schema-req

[XML Schema 5]
Schema for Object-Oriented XML 2.0
W3C Note 30 July 1999
http://www.w3.org/TR/NOTE-SOX/

[XML Schema 6]
Extensible Markup Language (XML)
Activity Statement
http://www.w3.org/XML/Activity.html

[XML DTD 1]
Datatypes for DTDs (DT4DTD) 1.0
W3C Note 13 January 2000
http://www.w3.org/TR/dt4dtd

[HTML 1]
HTML 4.01 Specification
W3C Recommendation 24 December 1999
http://www.w3.org/TR/html4/

[RF 1]
G. Ciocca, R. Schettini. A relevance feedback mechanism for content-based image retrieval, 1999.

[RF 2]
Ingemar J. Cox and Matthew L. Miller and Thomas P. Minka Thomas Papathomas, and Peter N. Yianilos – The Bayesian Image Retrieval System, PicHunter: Theory, Implementation and Psychophysical Experiments

[XMLDB 1]
Relational DB for XML Documents
Shanmugasundaram, Tuftc, Hc, Zhang, DcWitt, Naughton VLDB’99

[NN-search 1]
Content-Based Image Indexing

Tzi-cker Chiueh, Computer Science Department, State University of New York at Stony Brook, Stony Brook, NY 11794-4400 chiueh@cs.sunysb.edu

[NN-search 2]
The A-tree: An Index Structure for High-Dimensional Spaces Using Relative Approximation
Yasushi Sakurai† Masatoshi Yoshikawa§ Shunsuke Uemura§ Haruhiko Kojima†
†NTT Cyber Solutions Laboratories sakurai@marsh.hil.ntt.co.jp, kojima@aether.hil.ntt.co.jp ,Graduate School of Information Science
Nara Institute of Science and Technology {yosikawa, uemura }@is.aist-nara.ac.jp

[NN-search 3]
An Approximation-Based Data Structure for Similarity Search
By Roger Weber, Stephen Blott
Institute of Information Systems, ETH Zentrun, 8092 Zurich, Switzerland, weber@inf.ethz.ch
Bell Laboratories (Lucent Technologies) 700 Mountain Ave, Murray Hill, NJ 07974, USA, blott@research.bell-labs.com

[NN-search 4]
A Guttman ‘R-trees a dynamic mdex structure for spatial searching’, Proc ACM SIGMOD Int Conf on Management of Data, 47-57, 1984