OMMERCE ALERTER FOR WEB INFORMATION CHANGES

&

SEMI AUTOMATIC WEB WRAPPER

University of Essex

Henry George

MSc in E-Commerce Technology

DEPARTMENT OF COMPUTER SCIENCE

2004-2005

SURNAME:	George
OTHER NAMES	Henry
QUALIFICATION SOUGHT	MSc in E-commerce Technology
TITLE OF PROJECT	E-Commerce Alerter for Web Information Changes & Semi-Automatic Web Wrapper
SUPERVISOR	Jerome Robinson
DATE	16th September 2005

Abstract

Data extraction is an area of computer science that will come to play an increasingly important role in the near future. This project provides two separate software applications that can help in the understanding of the various aspects of data extraction and the analysis of the different problems and tentative solutions. First is the E-Commerce alerter system, a program which is developed to monitor the change of products on an E-commerce site (www.eBay.com Auctions). The program extracts items from the www.eBay.com site and stores them in a Relational database. It is scheduled to check for updates in information on the site every twelve hours. Also, it monitors the stored data for the purpose of reporting updates to the users who request them. If any changes are found, the user is informed via electronic mail. Secondly, a semi automatic wrapper has been developed, which is a tool to assist system developer to wrap HTML pages into XML Documents. The tool helps in extracting the items of interest from an html source page. The extracted data can be used by another application because it is stored in XML format, which is well structured. The technique used in developing semi automatic wrapper is tag Set Progression Grid (TpGrid) which is the fingerprint representation of an html page.

ACKNOWLEDGEMENT

I would like to thank all who help me to accomplish my dissertation. especially Mr. Jerome Robinson who served as my supervisor. He encouraged, gave advice, and guided me throughout the dissertation period. Also I would like to thanks Dr John Qiang Gan Module supervisor and all department members for their effort in facilitating and providing environment which was conducive for studying and doing a research.

ACKNOWLEDGEMENT

I would like to thank all who help me to accomplish my dissertation. especially Mr. Jerome Robinson who served as my supervisor. He encouraged. gave advice. and guided me throughout the dissertation period. Also I would like to thanks Dr John Qiang Gan Module supervisor and all department members for their effort in facilitating and providing environment which was conducive for studying and doing a research.

1. INTRODUCTION

The World Wide Web (www) is the one of the most useful tools to deliver information in the world. Many organisations use it for business purposes and scientific research purposes. The influx of new information in www is tremendous which makes difficult to get the right information for correct decision making.

One important point is the need to monitor changes of information in pages. Usually people are interested in the new information which appears on the page and not the old. Searching for new information in www from all relevant pages is time consuming and tedious work. The task of searching for new information has to be repeated as required even though no changes may be found.

1.1 Project motivation

The motivation in this project is to solve the challenging problem for data extraction and web monitoring. The problem of data extraction from web pages is still a crucial issue to solve since many researchers have tried to solve it, albeit with limited success. I am interested in investigating the different approaches used for data extraction from web pages, and the subsequent development of a warehouse such as a data-store to store the extracted data.

The monitoring of web data is also interesting since many people are interested to have a system which can keep an eye on data changes on the web pages and inform the interested parties according to their preferences. The system which I am intending to deliver will use a data warehouse to monitor pages instead of monitoring pages directly. It means that the data warehouse will be updated if and when there are alterations in the web pages being monitored.

Furthermore, the project aims to deliver a semi automatic web wrapper which will assist people in the extraction of meaningful information from the web pages.

1.2 Project Objectives

There are three objectives of the project: the first objective is to investigate different techniques used for data extraction from semi structured web documents. The second objective is to develop a tool which could be used to deliver alerts for information changes on web pages facilitated by Data warehouse technique to monitor updates and alterations. This goal is especially directed to monitor changes in Electronic Commerce web sites like www.ebay.com. The Third objective is to deliver a semi automatic wrapper for web data extraction. The semi automatic wrapper helps the user to select and filter the kind of data they want to extract from HTML source code. The tool will present the extracted data in XML format which will help other applications to have access to the collected data easily.

1.3 Application Domain

The areas where such a system is particularly in demand are:

The Stock Market

People in the financial world are greatly interested in monitoring financial news sites and/or tracking investment information. Many people are interested to know the trend of the market so that they can decide where and when to invest.

Sales and Marketing

The system can be used to monitor the performance and undertakings of competitors from business news sites. When a rival introduces a new product, one can immediately be informed.

Auctions

Track for item in bidding or selling and be notified when the price reaches a certain value. The people who want to monitor auctions may have no access to the original

source of data displayed on the web. So they a need tool which will assist them to monitor the progress of the auctions.

Price comparison tool

In writing program for price comparison, we need a program which can be used for page analysis as well as data extraction. The task of writing application for price comparison can be archived by the help of semi automatic wrapper. The supervised web wrapper helps to identify the interested clusters and data items.

1.4 Organisation of the dissertation

In the chapters that follow we will discuss in brief the contents of chapters of the dissertation. The dissertation has been divided into seven chapters.

Chapter 1: It explicates the introduction, project motivations and the objectives. Apart from motivations and objectives, the chapter describes the usefulness and applicability of data extraction and information monitoring.

Chapter 2: The chapter discuss the background of data extraction and alerter systems. It explains some research papers on information extraction from web pages. Five existing systems for web wrapper generation are also reviewed. It also explains the RSS technology for alerter systems.

Chapter 3: The chapter explains some terminologies used in web data extraction and data warehousing. It gives the functional as well as non-functional requirements for the alerter system. The chapter explains page analysis using Tag Set progressing Grid approach. The algorithm for data extraction is explained by the use of an example. Then the generalised algorithm for data extraction using Tag Set progression grid is given. Finally the chapter outlines the functional requirements for semi automatic wrapper generation.

Chapter 4: The chapter describes the design of the system giving the main components involved in the system. The chapter explains the general component architecture for the alerter system as well as the algorithm used by the alerter to extract information from

www.eBay.com web site. The chapter also explains the general component architecture for semi automatic web wrapper. The algorithm used by the semi automatic web wrapper to define data schema is described. Lastly the description of technology used to implement the project and the tools required for the system to work.

Chapter 5: The chapter starts by describing the classes involved in the alerter system and the relationship between classes for each component in the alerter system. The semi automatic wrapper classes are also described. particularly the ones responsible for the implementation of the algorithm used by the wrapper to define data schema. Finally the chapter describes and shows the interfaces used for both the alerter system and semi automatic wrapper program. In general the chapter portrays the system implementation.

Chapter 6: It describes number of tests carried out in the alerter system. It describes Recall and Precision as a method used to measure the success of wrapper in data extraction. The chapter describes by giving sample page where semi automatic wrapper was able to extract data successfully.

Chapter 7: The chapter describes the achievements. problems encountered. future work for semi automatic wrapper and project management.

2. BACKGROUND

2.1 Information extraction from web sources.

The Electronic alerter system requires the information to be extracted from the web pages so the extracted data is used for the purpose of monitoring updates from the web pages.

In reviewing the vast and promising field of information extraction and alerter systems, some valuable references were found which present different methodologies for data extraction and detection of changes on web pages. It was discovered that web changes may take two forms: structural change and information change. In order to extract meaningful information from web pages these two forms for changes must be addressed.

In their paper Detecting and delivery information changes on the web: webcq, Calton Pu, Ling Liu, Wei Tang [6] focuses on the use of structure present in Hypertext and the concept of continual queries. It describes the mechanism to discover and detect changes in the World Wide Web. It provides a tool which consists of four components. The components are described below.

The system has four components. The first component is Change Detection Robot, which detects changes on the pages. The second component is a proxy cache service. The aim of this module is to reduce traffic from the Internet. The static pages are stored in the cache. The third component is Personalised presentation tool, which highlights changes detected by the WbCQ sentinels. The fourth component is change Notification service, which delivers fresh Notification to the appropriate user.

The paper Efficient Web Change Monitoring with Page Digest, by David Buttler, Daniel Rocco, Ling Liu [8] deals with the techniques required for data extraction and the algorithms which are used to improve extraction efficiency. It addresses the problem by grouping multiple related queries in a single process for the purpose of reducing network overload.

Bing Tan, Schubert Foo, Siu Cheung Hui in their paper. Web information monitoring: an analysis of web page updates [18] proposes the idea of studying the rate of changes of pages before writing a program to extract data. The analysis results which are obtained from the study will be used to define functions and features for a web monitoring system. The paper [8] shows that the text, hyperlink, and last updated date are features to be used for monitoring information changes from the web pages.

In addition to information changes from the pages the paper by Line Eikvil [11] makes the point that not only information can change but also the structure of the page may change. That means in addressing the problem of data extraction both changes must be addressed. The possibility of change in web page structure is high, therefore in the work of writing a web wrapper for data extraction from the web pages, the issue of monitoring page structure change must be given due importance. If a wrapper is written and the page structure changes then the wrapper will not work. To avoid the extraction of meaningless data, we have to use techniques which will test the web page structure before the data extraction can be done. The paper [4] gives the techniques for analysing web pages before writing wrapper for data extraction. The major issue, according to this paper, is to identify repetitive patterns of tag Sets, which will be used to locate the area which is rich in data items as well to test the page before data extraction.

There are many papers describing different approaches to identifying and extracting information from web pages. However, for the development of semi automatic web wrapper for this project, the data extractor will be based on the technique of finding repetitive patterns of tag Sets to identify the area which is rich in data items.

2.2 Sample systems for wrapper generation.

The web wrapper can be defined as an application which can be used to extract meaningful information from web sources. There are three approaches that may be used to generate wrappers. These approaches are, manually coded, semi automatic and automatic approach.

Generating a wrapper manually is not a difficult task, since it requires the developer to understand the structure of HTML source (web page) and to find the landmarks for the program to mark were to find the data items. The landmarks can be html tags or sequence of text strings. The wrapper produced by hand has a great risk of high maintenance cost. It is very likely that upon the slightest change in the page structure, the wrapper may fail to extract data, causing the need for the development of a new wrapper.

Semi automatic wrapper generation requires supervision of a person before the extraction of data. The wrapper has to be trained to understand what fields to extract. The semi automatic approach is less error prone and easy to maintain than hand coded approach.

Automatic wrapper generation uses machine learning techniques. The automatic wrapper generation systems require two phases, the learning phase and extraction phase. The system uses sample pages to learn before starting the task of data extraction.

The following systems have been developed for data extractions from web sources. (SHOPBOT, WIEN, Soft Mealy, RAPIER, STALKER).

2.2.1 ShopBot

ShopBot [1] is a comparison shopping agent specializing in the extraction of information from different web sites (especially for online stores). The prime aim of the agent is to undertake price comparison. The algorithm used by the agent assumes that the information displayed on the web pages are in a repetitive structure like a table with columns and records. The methods used by the agent to extract information include a combination of heuristic search, pattern matching and inductive learning techniques.

The ShopBot agent needs two phases to operate. The first phase is the learning phase, which is performed offline. The second phase is comparison shopping, which is performed online.

In the learning phase, the specific site is analysed to get symbolic description of items in the site. The descriptions are used to extract data from the sites which can be used for price comparison in the second phase.

The learning phase use heuristics to find the appropriate search form. The learner determines the format of the result page. The resulting page for the query is assumed to be composed of header, a body and a tailer. The body contains all the desired information. The assumption is taken that the header and tailer are consistent across different pages.

In analysing the page format, the algorithm uses three steps to determine the format. The first step is to determine failure template. Non-existing products are identified by analysing the result page by querying for dummy products. Dummy products are the products which do not exist in the online vendor database. e.g. "radioXXXXXXX". For sure if dummy product is searched the result page will not display products. So, dummy products are employed to determine the failure template.

The second step is to determine a successful template. The learner performs several queries for popular products and matches the results with failure template. The learner records the generalized templates for the header and tailer of successful results pages.

Finally, the body containing the product description is determined. The algorithm assumes the HTML pages consist of only tags and text. The format of product description is modelled as a sequence of HTML tags and text strings. The algorithm assumes every product description starts with a new line by using HTML tag (i.e. <p> <b> etc). The agent makes assumption that products displayed as query result on the search form are generated by the sample template in the server so they have similar appearance. In so doing the body could be divided into logical lines representing vertical-space delimited text. Then the algorithm compares the different abstract formats and logical lines to find the best match. When the product description format is identified the price is extracted by using hand coded web wrapper.

2.2.2 Wrapper induction environment (WIEN) application

WIEN [19] is a tool developed for assisting the construction of wrappers. It is the one of the first tools used to automate wrapper production. WIEN employs learning induction for wrapper production. Wrappers produced using learning induction techniques can work for semi–structured data as well as structured documents.

The WEIN algorithm assumes that web document conforms to HLRT organization, where H stands for head delimiter, R is a set of Right delimiters, and L is a set of left delimiters for the facts from the web documents and T is a Tail that denotes end.

The induction algorithm takes a sample of web pages from a given resources to produce appropriate classes which will be used later for data extraction. It needs sufficient sample documents to reduce the probability of wrapper failure.

The technique used in WIEN, by taking consideration that only delimiters immediately followed by data items to be extracted works for only well formed sites. The approach cannot handle the problem of missing or varying data items.

2.2.3 SoftMeal

SoftMeal [12] is a system that learns to extract data from semi structured web sources by learning wrappers specified as non deterministic finite automata. It uses inductive general algorithm to induce contextual rules from training examples. The trained examples return an ordered list of the items to be extracted. The items are separated by symbols which act as the separator between items.

2.2.4 Stalker

"Stalker [20, 11] is a supervised learning algorithm for inducing extraction rules. The training samples are supplied by the user, who has to select a few web sources and mark up the relevant data.

The algorithm uses sequence of tokens to mark up the start of data items. The landmarks can be words or HTML 'TAGS'. A landmark is used to locate data items to be extracted on a page. STALKER models HTML web page as a tree structure. The extraction uses a breadth first approach where all the nodes at similar levels are extracted together. The algorithm can handle missing data items.

2.2.5 Rapier

RAPIER [16, 11] is the Robust Automated production of Information rules. The system is used to extract information from semi structured text as well as structured text. It takes a pairs of documents and filled templates indicating the information to be extracted. The pattern matching rules are used to extract the relevant data items. RAPIER needs human supervision to be able to extract data. the human feeds the system with text and filled templates.

2.3 Alerter Systems.

Alerter systems [21] are programs that check automatically the web pages and detect changes in their content. The programs are used to keep track of the element of interest to a client. When the items of interest found or changed on the web page, the alerter immediately notifies the client who has interest on it. The most popular technology which is used for web sharing and monitoring is known as RSS. The description of RSS is followed in this section.

What is RSS: - RSS is the abbreviation for Really Simple Syndication. It is the way of broadcasting news to vast numbers of people on the web. RSS is an XML based format which allows sharing of information on the web. The RSS technology provides the summary of news to the client and the link to the main body of the summary. If the client is interested to the summary then the decision is left to the client to decide to follow the link provided.

People who surf the web are interested in websites whose contents change dynamically. They need current and up-to-date contents which may be relevant to them. So repeatedly

looking on each website to see if there is any new content can be very boring work. By using RSS, the client computers fetch the summary, updates, links, and headlines on behalf of the user from various web sites.

To some extent RSS is one of the better way for notification of new and changed content from the web pages. It could be used for notification of the arrival of new products in a store, alerts changes of web content and notification of addition of new item to database store.

The RSS as alerter system it works as follows, the webmaster maintains an RSS feed, or channel. The channel is RSS file which is normally saved on the server. The Client machine can regularly fetch the file to get the most recent items. Because RSS is like XML document, we need a program to read the document. The program which is used to read RSS is called Aggregator. The Aggregators offer a number of functions such as integrating several related feeds into a single view and hiding those items which have been already seen.

The alerter system which is provided by RSS may be limited to the amount of information which may be needed in the electronic commerce field. Normally the RSS contains headline, description and the universal resource locator (URL). In that case RSS will not tell the detailed description of items like price. The RSS feeds may contain unsatisfactory information, information may be too little. Another deficiency comes to the fore, when for example: the applications need to use data from RSS for shopping price comparison. It will be impossible since the gathered information may not include price of items. Furthermore, the RSS feeds are controlled by the owner of the sites and some sites do not provide RSS feeds at all.

2.4 Summary

In this chapter we pinpoint three key areas: The first area is on the literature review on the information extraction, as the E commerce alerter system is concern to data extraction and dispatching the extracted data. Secondly we described the existing systems for data

extraction from the web pages .Finally the RSS technology as the way of broadcasting news and the limitations imposed on them.

3. SYSTEM ANALYSIS AND REQUIREMENT

The project is divided into two parts. One focuses on the electronic alerter system and another part aims to undertake semi automatic wrapper generation. The approach adopted for the web data monitoring (alerter system) is data warehousing. The interested data items are fetched from the preferred web data source and then stored in a relational database. The program is triggered to extract data items after uniform intervals of time. When information is required to be monitored, it will be easy to detect changes from the database rather than using web page directly. The technique used for data extraction in the first part of the project is hand coded wrapper since the web source selected for the implementation of the project uses java script to display its information. The second part of the project which is a semi automatic wrapper generation uses the tag Set Progression Grid (TpGrid) [4] for page analysis.

3.1 GENERAL TERMS

3.1.1 Data Warehouse

A Data warehouse is a repository of collected data from different sources. The collected data in the warehousing can be used for different purposes. For example it can be used to answer different queries as well as monitoring updates from the original source. The web pages are considered as data source for the Data Warehouse. The architecture of Data Warehouse is shown in Figure 1.

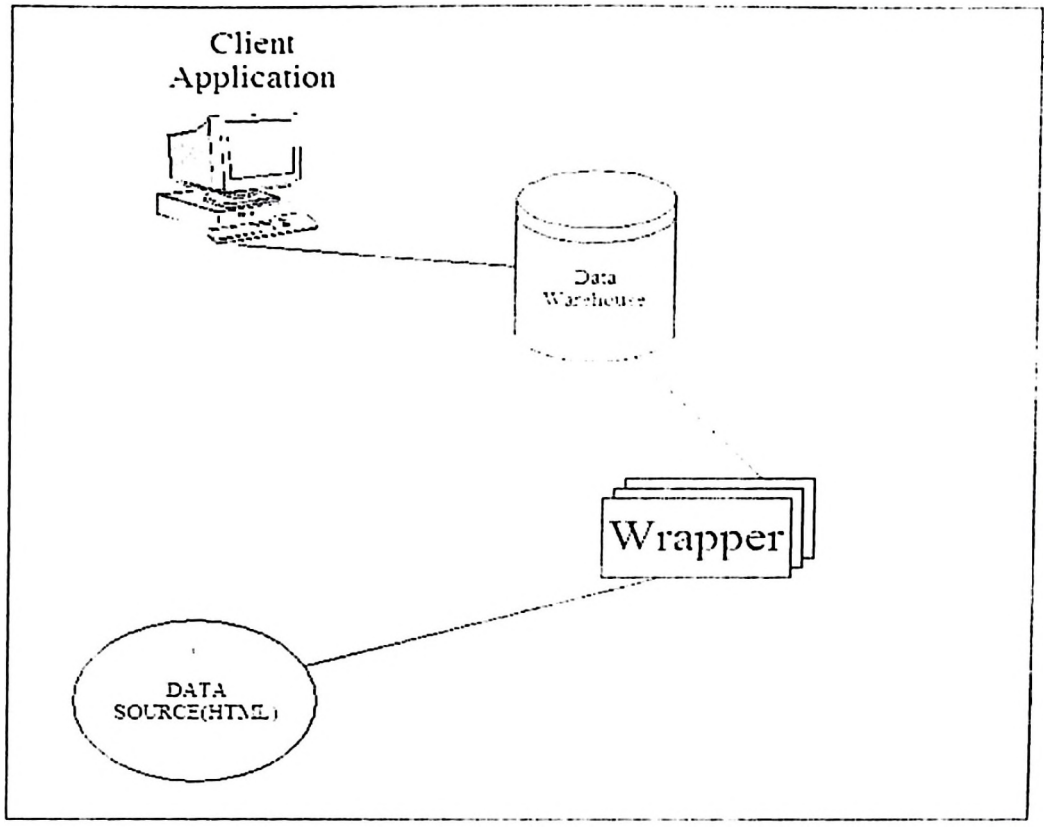


Figure 1: Data Warehouse Architecture

3.1.2 Information Extraction

The aim of information extraction (IE) from web sources is to transform the semi structured documents into structured documents. Structured document is a document presented in tabular form whereby a machine application can use that data. It is easy for the machine to use information which is presented in the tabular format rather then in a semi structured form. One of the techniques used to extract relevant information from the web sources is by the use of wrappers.

3.1.3 Structured data

Structured data is data which is normally stored in traditional databases and presented in tabular format or in the extensible mark language (XML) as data centric documents. The information presented in tabular format can be readily used by a machine application.

Manipulation of information presented in a tabular format can be done through simple techniques like Structural Query Language (SQL).

3.1.4 Semi structured data

Semi structured data is the opposite of structured data, in the sense that it lacks regular structure. The information on the web pages is normally in semi structured format. Some of the information displayed on the page is generated by the same template from the server, so the appearance of the page looks similar.

The properties associated with the information displayed on a page are records with missing values, multiple values and nested items. When extracting information from the pages we need to address these properties in order to maintain the semantics of the extracted data items.

3.2 ELECTRONIC COMMERCE ALERTER SYSTEM

Electronic mail system is the core communication system which can be used for reporting changes in the web contents. For example, in auctions when an item reaches a certain level of price, an alert can be sent across the Internet to warn the person who is interested in it to see the changes. Another scenario could be envisioned if a Data warehouse is used to store the collected data from web pages, then the email system can be triggered to send a summary of changes that occurs from the monitored products after a specified period of time.

3.2.1 Functional requirements for electronic commerce alerter system

1. The system should be able to extract data from web pages (www.eBay.com site) and store the meaningful data to a relational database (MySQL Database).
2. The system should provide search functionality from the local database (local cache)
3. The system should process web pages and send the extracted results to those who are interested to view that information in an organized fashion.

4. The system should provide the web page monitoring functionality and report the changes which may appear.
5. The system should provide the automatic updating of Relational database by extracting new updates from the web pages. It should provide scanning of web sites for changes in data by making a comparison to previous values stored in the database.

3.2.2 Non -Functional requirements for Electronic Commerce alerter system

1. Security requirement: The new client should be able to create a new account to use the system.
2. Operational requirements: The system should run on Tomcat server
3. Performance requirements: Database used should be able to hold large volume of data and provide efficient concurrent access.
4. External requirements: The system shall provide interface that allows any user to alter the data he/she wants to monitor.

3.2.3 Use case Diagram

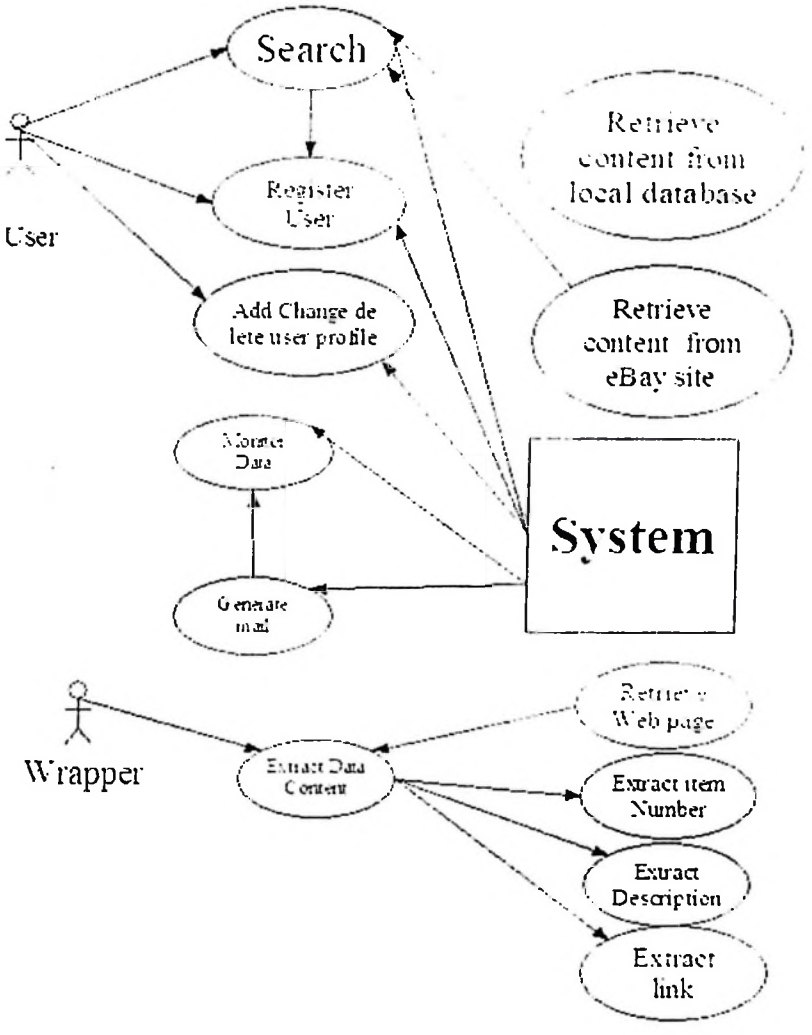


Figure 2: Use case Diagram for electronic commerce alerter system.

3.3 SEMI AUTOMATIC WRAPPER GENERATION

The second part of the project is to generate a semi automatic wrapper for web data sources. A wrapper is considered as a key component for data extraction. The web wrapper can be defined as an application that is designed to extract meaningful information from web sources.

The wrapper accepts queries and extracts data items from web sources. The basic task of wrapper is to fetch and extract relevant data items according to the query provided. For the wrapper to extract meaningful data, it needs a set of rules to follow to achieve the objectives of extracting correct data. In theory wrapper is supposed to be able to download HTML source pages, extract meaningful data and then store them in structured format.

The nature of web page source presents some problems such as the data not being of regular structure, structure changing dynamically, pages may include missing field value or attributes in the tabular presentation on the page etc. Also information on the page may not follow the sequence ordering like relational database tables. A field on the page table may contain more than one value, so when designing a wrapper these problems must be addressed. The semi automatic wrapper generation requires supervision of humans to extract meaningful data by putting some rules for wrapper to follow.

3.3.1 Semi automatic wrapper page analysis

Definition of some terminology used:-

HTML is an abbreviation of Hyper Text Mark-up Language. It is a language used to display information on the page. A sample HTML page is shown in figure 2.

```
<HTML><HEAD> <TITLE>TITLE OF PAGE
</TITLE></HEAD> <BODY>THIS IS MY FIRST HOMEPAGE.
<P>THIS IS NEW PAGE
</P></BODY></HTML>
```

Figure 3: Simple HTML page

According to Jerome Robinson (2004), the definitions for some sub-string on HTML document are stated below.

Tag is a command which tells the web browser how to display the HTML content. In figure 3. <P> is a tag which tell the browser to display text in new paragraph.

TextString is a non white space character which is not enclosed in an angled bracket. example "THIS IS NEW PAGE" in figure 3.

TagString. A tagString is a sequence of HTML tags that occur before the text string. for example in figure 3. <HTML><HEAD><TITLE> is a tagString.

TagSet is a set of tag names with a count value for each name that occurs in the corresponding tagString. For example from figure 3. the whole document contains the following tag names "HTML. HEAD. TITLE. /TITLE. /HEAD. BODY. P. /P. /BODY. /HTML."

The document contains the following tagString:-

TagString 0: <HTML><HEAD><TITLE>

TagString 1: </TITLE></HEAD><BODY>

TagString 2: <P>

TagString 3: </P></BODY></HTML>

```
<HTML><HEAD> <TITLE>TITLE OF PAGE
</TITLE></HEAD> <BODY>THIS IS MY FIRST HOMEPAGE.
<P>THIS IS NEW PAGE
</P></BODY></HTML>
```

Figure 3: Simple HTML page

According to Jerome Robinson (2004), the definitions for some sub-string on HTML document are stated below.

Tag is a command which tells the web browser how to display the HTML content. In figure 3, <P> is a tag which tell the browser to display text in new paragraph.

TextString is a non white space character which is not enclosed in an angled bracket, example "THIS IS NEW PAGE" in figure 3.

TagString, A tagString is a sequence of HTML tags that occur before the text string, for example in figure 3, <HTML><HEAD><TITLE> is a tagString.

TagSet is a set of tag names with a count value for each name that occurs in the corresponding tagString. For example from figure 3, the whole document contains the following tag names "HTML, HEAD, TITLE, /TITLE, /HEAD, BODY, P, /P, /BODY, /HTML."

The document contains the following tagString:-

TagString 0: <HTML><HEAD><TITLE>

TagString 1: </TITLE></HEAD><BODY>

TagString 2: <P>

TagString 3: </P></BODY></HTML>

TagString is represented in TagSet format with count values as '

<1-HTML><1-HEAD><1-TITLE><>0-/TITLE. 0-/HEAD. 0- BODY. 0- P. 0-/P. 0-/BODY. 0-/HTML'

TagSet TagString 0: can be represented in the vector format as

TagSet0: 1.1.1.0.0.0.0.0.0.0.

Writing a web wrapper to extract data from web pages requires two steps. The first step is web page analysis. It's important because it produces a suitable web page descriptor which will be used by the extractor to locate and extract data in the html raw source. There are several ways whereby the source code of page can be analysed.

For example the page can be analysed using raw html code. Looking for HTML tags in the whole document the approach treats an html document as an XML document. Since the html tags are not well formed like XML then this approach is difficult to use. Another approach is to use tag set progression Grid (TpGrip) representation of the web page. Analysis looks for the repetition pattern of sets of tagsets in the web page document. The approach is much simpler because it gives the fingerprint of the page representation.

The second step is to write the actual program which extracts data. The program uses the descriptor obtained from step one. The descriptor is the fingerprint which the program will use to locate the relevant data items.

Page analysis

In the production of semi automatic wrapper, we need to produce the program which will be used to analyse the page and gives the possible clusters present in the HTML document. The approach used to analyse web page source code is tag set progression grid representation of web page. The assumption taken is the HTML document is composed of tags and text items. Information in the web page is enclosed between tags. No Data is

The presentation of figure 4. is represented in the sequence of numbered (tagSet. textString) pairs which form the tagSet Progression Grid as shown in Figure 6.

```
 0  tagSet0: 0
 1  tagSet1: 1
 2. tagSet2: 2,4,6,8,10,202,204,214,216,218
 3  tagSet3: 3,5,7,9,11
 4  tagSet12: 12
 5. tagSet13: 13
 6. tagSet14: 14
 7. tagSet15: 15,16,17,18
 8. tagSet19: 19
 9. tagSet20: 20
10  tagSet21. 21,22,23,24,25,26,27,28,29,30,31,32,33,34,35,36,37,38,39,40,43,44,45,46,47,48,49,50,51
11. tagSet41: 41
12. tagSet42: 42
13. tagSet52: 52
14. tagSet53. 53
15. tagSet54: 54
16. tagSet55: 55
17  tagSet56. 56,58,215,217,219
18  tagSet57: 57
19  tagSet59: 59

20  tagSet60: 60
21  tagSet61: 61,68,75,82,89,96,103,110,117,124,131,138,145,152,159,166,173,180,187,194
22  tagSet62. 62,69,76,83,90,97,104,111,118,125,132,139,146,153,160,167,174,181,188,195
23  tagSet63: 63,70,77,84,91,98,105,112,119,126,133,140,147,154,161,168,175,182,189,196
24. tagSet64: 64,71,78,85,92,99,106,113,120,127,134,141,148,155,162,169,176,183,190,197
25. tagSet65: 65,72,79,86,93,100,107,114,121,128,135,142,149,156,163,170,177,184,191,198
25  tagSet66  66,73,80,87,94,101,108,115,122,129,136,143,150,157,164,171,178,185,192,199
27. tagSet67: 67,74,81,88,95,102,109,116,123,130,137,144,151,158,165,172,179,186,193
29  tagSet200. 200

30. tagSet201: 201,203
31. tagSet205: 205
32  tagSet206: 206
33  tagSet207: 207
34. tagSet208: 208,209
35. tagSet210. 210
36  tagSet211  211
37  tagSet212: 212
38. tagSet213: 213
39. tagSet220. 220
40. tagSet221: 221
```

Figure 6: The appearance of Tag Set Progression Grid (TpGrid)

TagSet 61 to TagSet 67 contains long rows. so tagSet/textString pairs from 68 to 199 contain tagSets already used. The new tagSet after 199 is tagSet number 200. The structure formed is called tagSet Progression Grid (TpGrid). The row cluster tagSet 61. tagSet62. tagSet 63. tagSet64. tagSet 65. tagSet66. tagSet 67 shows the positions of seven field result records in the web page. The extractor program can use the structure obtained to extract data accurately.

embedded in the client script language such as Java script. In short we can say the HTML document is modelled as tagString/textString pairs in its html source code.

For the case study in the real world, we are analysing one page from "http://www.halfpricecomputerbooks.com" web page. The search was provided to the site with key word database, the result was displayed by the browser as shown in figure 4.

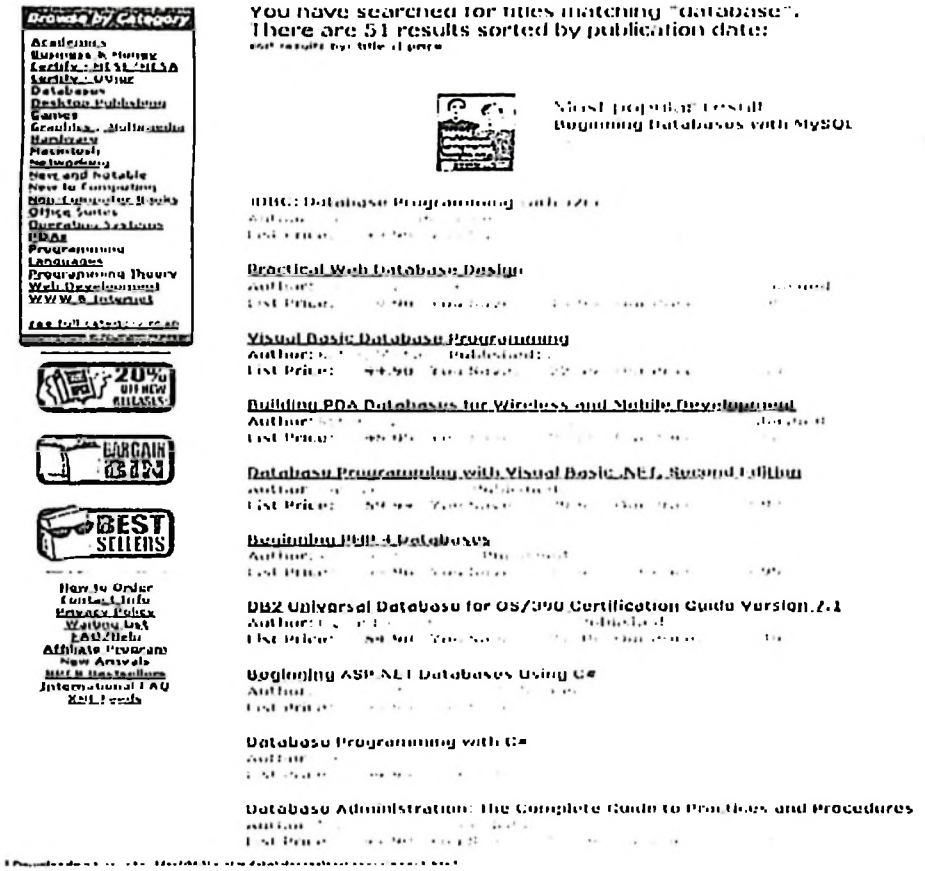


Figure 4: Results of the query of the keyword database to http://www.halfpricecomputerbooks.com/.

In reference to figure 4, the whole document contains the following Tag name. HTML, HEAD, LINK, TITLE, /TITLE, META, /HEAD, BODY, table, tr, td, a, img, /a, /td, div, /tr, b, font, /font, /b, /div, FORM, INPUT, SELECT, option, /option, /select, /table, br, /FORM, strike, /strike, p, ul, li, /li, /ul, center, /center, /body, /html.

The number of text Strings in the document in figure 4 were counted and found to be 220. The number of tagSets found in figure 4 is 220. So there are 220 tagSet/TextString pairs. The tagsets in the document were analysed and the duplicates were removed. Then the number of distinct tagsets found was 40: this implies some of them were reused several times.

The HTML document in figure 4 is represented in the format of vector counts of tagSets as shown in Figure 5.

```
tagSet0   1,1,1,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
tagSet1   0,0,1,0,1,4,1,1,1,2,3,2,3,1,2,2,1,1,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
tagSet2   0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
tagSet3   0,0,0,0,0,0,0,0,0,0,0,1,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
tagSet12: 0,0,0,0,0,0,0,0,0,1,1,0,0,1,1,1,1,1,1,1,1,1,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
tagSet13: 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
tagSet14: 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
tagSet15: 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
tagSet19: 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,1,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0
tagSet20: 0,0,0,0,0,0,0,0,3,4,4,1,1,0,2,0,2,1,1,1,1,1,0,0,0,0,0,0,1,1,1,0,0,0,0,0,0,0,0,0,0,0
tagSet21: 0,0,0,0,0,0,0,0,0,0,0,1,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0
tagSet41: 0,0,0,0,0,0,0,0,0,0,0,1,0,1,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,2,0,0,0,0,0,0,0,0,0,0,0,0
tagSet42: 0,0,0,0,0,0,0,0,0,1,1,4,6,4,3,0,3,1,1,1,0,0,0,0,0,0,0,0,2,7,0,0,0,0,0,0,0,0,0,0,0,0
tagSet52: 0,0,0,0,0,0,0,0,0,0,1,0,0,1,1,0,0,0,1,1,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0
tagSet53: 0,0,0,0,0,0,0,0,0,0,1,0,0,0,1,0,0,1,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
tagSet54: 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,1,1,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0
tagSet55: 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0
tagSet56: 0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
tagSet57: 0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
tagSet59: 0,0,0,0,0,0,0,0,0,0,0,1,0,1,0,0,0,1,1,1,1,0,0,0,0,0,0,0,0,3,0,0,0,0,0,0,0,0,0,0,0,0
tagSet60: 0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0
tagSet61: 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0
tagSet62. 0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0
tagSet63. 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0
tagSet64  0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
tagSet66  0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,1,1,1,0,0,0,0,0,0,0,0,0,2,0,0,0,0,0,0,0,0,0,0,0,0
tagSet179: 0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,2,0,0,0,0,0,0,0,0,0,0,0,0
tagSet199  0,0,0,0,0,0,0,0,0,0,0,2,1,1,0,0,0,1,1,0,0,0,0,0,0,0,0,0,0,2,0,0,0,0,0,0,0,0,0,0,0,0
tagSet201: 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
tagSet202: 0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,1,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
tagSet204: 0,0,0,0,0,0,0,0,0,0,1,1,1,1,1,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0
tagSet205: 0,0,0,0,0,0,0,0,0,0,1,0,0,0,1,0,0,2,2,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0
tagSet206: 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,1,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0
tagSet207: 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,1,1,0,0,0,0,0,0
tagSet209: 0,0,0,0,0,0,0,0,0,0,0,1,1,1,0,0,0,1,1,2,2,0,0,0,0,0,0,0,0,2,0,0,0,0,0,1,1,1,0,0,0,0
tagSet210: 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0
tagSet211: 0,0,0,0,0,0,0,0,0,0,0,1,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,2,0,0,0,0,0,0,0,0,0,0,0,0
tagSet212: 0,0,0,0,0,0,0,0,1,2,2,1,0,1,2,0,2,1,1,1,1,0,0,0,0,0,0,0,1,2,0,0,0,0,0,0,0,0,0,1,1,0
tagSet219: 0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,1,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
tagSet220: 0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,1,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,1,1,1
```

Figure 5: Distinct tagSet representation of figure 3 with count values of 42.

Figure 6 shows all text items visible in Figure 4. The items 61. 62. 63. 64. 65. 66. 67 have used the same tagSet as 68. 69. 70. 71. 72. 73. 74. The appearance on the browser is similar. The appearance looks similar for the records starting from the first record which is the record with tagsets (61. 62. 63. 64. 65. 66 and 67) to the last record which use tagsets (194.195.196.197.198 and 199).

The structure in figure 6 is a page descriptor which is used by the extractor to find and extract relevant data. The tagSet60 is marker for the start of records while tagSet 199 is the end of records. This is the end of page analysis for web source http://www.halfpricecomputerbooks.com/.

3.3.2The data extraction algorithm

The tag set progression grid in Figure 6 provides the details for the extractor program to identify data. The extractor work as follows. Search for tagSet 60 and mark this as the start of the data. The tagSet 60 is immediately followed by the data. TagSet 200 is used by the extractor to mark the end of extraction. If the extractor finds TagSet 200. it will immediately cease the extraction of information. After the extractor marks the starting and stopping points. it continues to extract data items as long as the specified TagSets are found.

3.3.3 Generalized data extraction algorithm

From figure 6. we can generalize the structure of tag set progression grid (TpGrid) to be used to extract data from web sources. The generalized Characteristics of row cluster in the tag set progression grid (TpGrid) are shown in Figure 7 and the explanation of the algorithm follows.

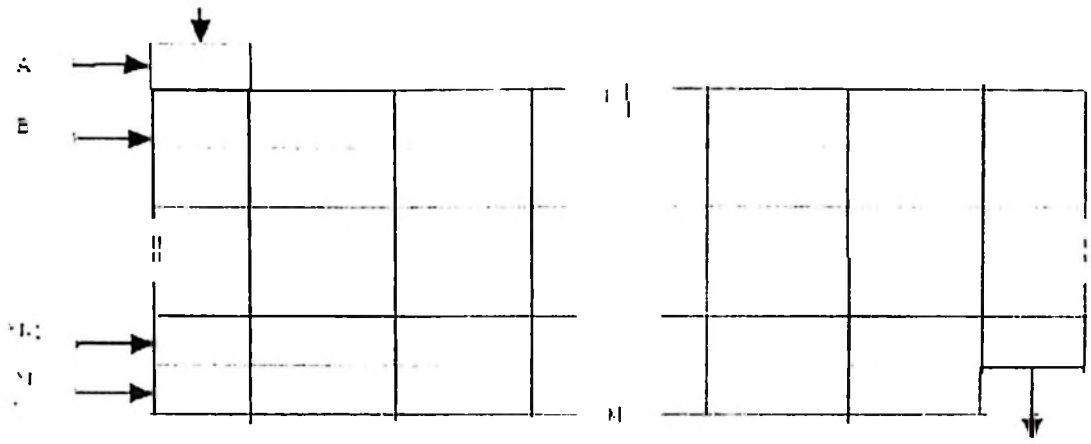


Figure 7: The characteristic structure of a result set row cluster in tag set progression grid (TpGrid).

The way in which data items are discovered according to TpGrid [4] is by following the consecutive item numbers found in the TpGrid. The trail is formed by following the repeated vertical sequences of N items. After every N item the structure repeats which makes it cyclic. The presence of a repeated cycle is the confirmation that the data items displayed on the Brower have the same appearance. Hence the same tagString can be used for display. From figure 7. tagSet A immediately is followed by data item. The first field in the first record is found before TagSet B. The last Field in the first record is found at tagSet N-1. The reason why the first field is not at tagSet B as expected is because the first tagString occurs only once before the big cluster is formed. For the second record up to the last the first field in each record is found at tagSet N and the second field is at TagSet B and the last field is at tagSet N-1. As shown in Figure 7. the entry to the cluster is at tagSet A and the exit is at tag Set N.

3.3.4 Functional requirements for semi automatic wrapper generation

1. The wrapper should accept HTML source code as a source of information.

2. The system should allow the user to define the schema and subsequently extract data according to the defined schema.

3. The wrapper should allow the extracted data to be saved in the storage media in the XML format.

The diagram 8 shows the use case diagram for semi automatic wrapper.

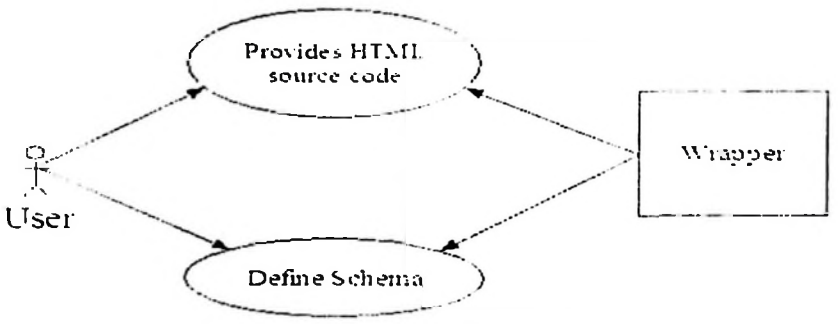


Figure 8: Use case for semi automatic wrapper.

3.4 Summary

The chapter start by describing the terminologies used in the field of data extraction from the web. Secondly the functional and non-functional requirement for both alerter and semi automatic web wrapper systems are provided. Thirdly the sample HTML page was analysed using new model called tag set progression grid and illustrating the algorithm used for data extraction using the tag set progression grid.

4. SYSTEM DESIGN

4.1 ALERTER SYSTEM

4.1.1 General components architecture

The rectangles represent the basic components of the system. The arrows represent the flow of data between components. Dotted lines stand for some internal part of a component.

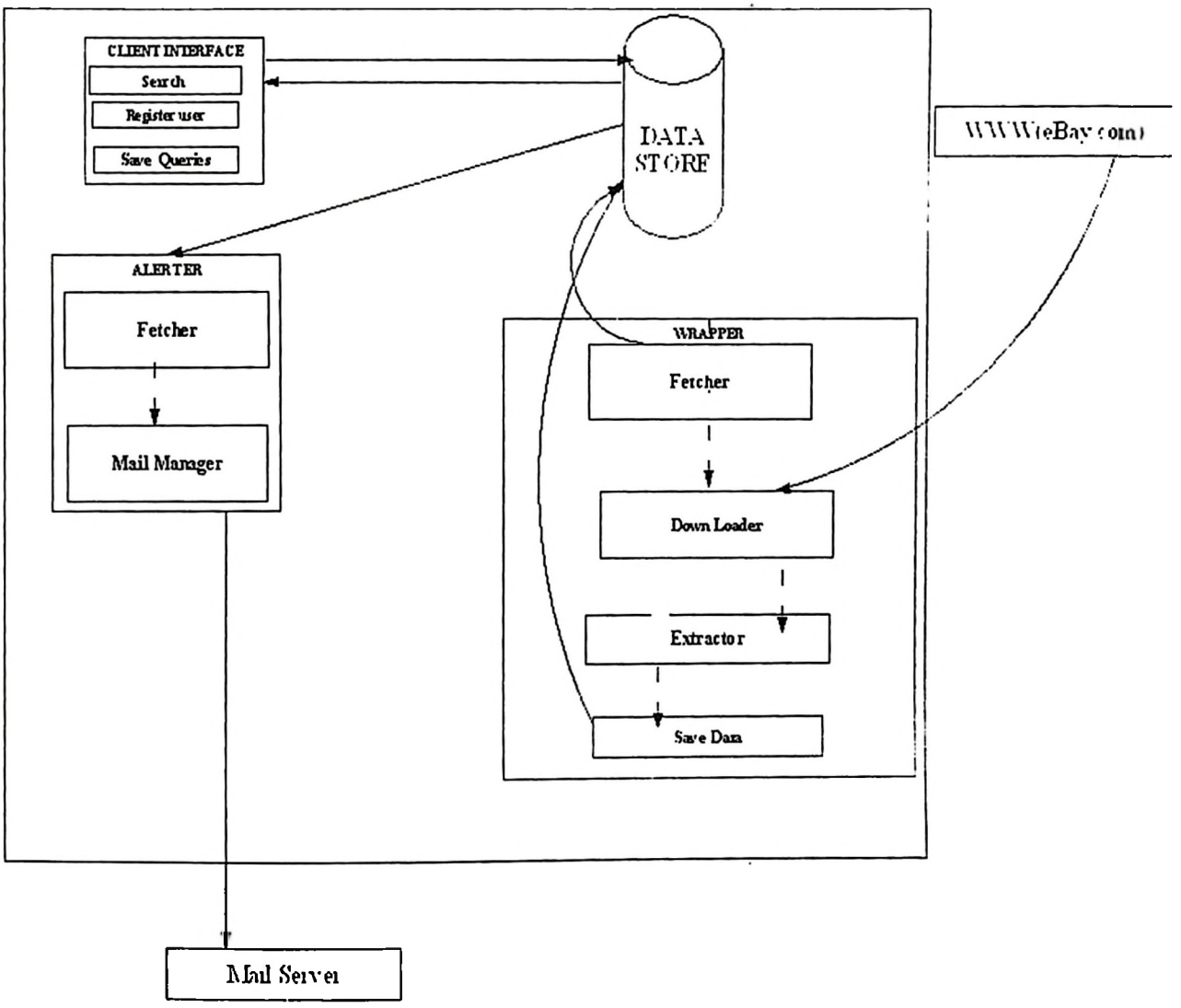


Figure 9: Alerter System components

The system has four components that are the client interface, local database (store), alerter and wrapper components.

General components are the components which perform the basic operations of the system. The services provided by each component are integrated to make the system functions as intended.

The general services provided by each component are described.

4.1.2 Client interface component

The component is responsible for providing services to the end user. It is responsible for collecting user inputs. The user inputs include search, registration, and taking the user queries. The input to the system is by keying on the keyboard.

The Client interface component is the web based graphical interface. Functions provided by the component are:

It provides the facility to the user to register to the system. It gives the user options to write the information they intend to monitor from the pages. It provides a mechanism whereby the registered users are required to login so as to have access to the system. Another task for client interface is to provide the ability of the user to alter the previously saved queries.

As we can see the main task of the client interface component is to collect user's interests and deliver them to Data store. It provides the interface for searching data from the Data store.

4.1.3 Data store

The Data store component serves client interface, alerter and wrapper components.

Firstly, the local database accepts the inputs collected from the user interface and make them persistent. It answers queries asked by the client interface, that is to say when users are searching extracted data they get the results from the Data store.

Secondly, the Data store serves the alerter component. Alerter component regularly visit the local database for updates.

Thirdly, the data extracted from the web is stored in the Data store. Extractor takes the saved queries from local database, and then downloads web pages from World Wide Web (www.ebay.com). The extracted data is stored in the local database. In short, the local database act as a data warehouse for extracted web data.

4.1.4 Wrapper component

The wrapper component is composed of fetcher, downloader and extractor subcomponents. Fetcher is responsible for requesting user queries from local database. All queries for all users are taken one by one. The downloader accepts the queries from fetcher and then downloads pages from World Wide Web (www.ebay.com). The downloader sends the downloaded page to Extractor component where the actual extraction is done.

The Extractor is responsible for finding relevant data from the downloaded pages. Extractor makes a comparison between already extracted data from local database and that extracted by the downloader. If new data is found then the extractor sends the new data to local database.

Downloading pages from World Wide Web is resource consuming (time and memory). In order to cope with this problem and avoid the slowing down of the system and preventing memory problems, the downloader is scheduled to download pages twice a day. It means the downloader is designed to download pages after every twelve hours.

4.1.5 Alerter component

The alerter component is responsible for sending notification to the user. The alerter sends the notification in the form of text. The alerter is responsible for finding new data extracted from the pages. Alerter fetches changes from the local database component. The alerter component can summarize all changes requested by a particular user and sends the result to mail server. Every hour alerter is scheduled to fetch updates from the local database. But it's not necessary that every hour the changes found in the local database are notified to the user. In the client interface, users are requested to specify the time which is appropriate for them to receive notifications.

For the user to receive notifications they must be registered. The system sends mail user to ask if they are ready to accept notifications from the system. For the users to receive to notifications they must accept by clicking on the universal resource locator (URL) and then logging in to the system.

4.1.6 Web Extractor

It is easy to write a program to query a web page from a web server. It is easy because the web page is machine readable meaning that the page is a string of text characters. The difficulties are to recognize which piece of text string represents the data items. Sub-strings within the html document represent data, but these sub-strings are embedded in the scripts or html codes.

EBay Web site (http://www.ebay.com/):

EBay pages are analysed by looking for sub-strings which represent the items. The sub-strings are marked as landmarks for the extractor program. The extractor uses the landmarks to find the data from the EBay pages.

Analysis of EBay web sources

Originally my project proposed the use of tag set progression (TpGrid) for page analysis on EBay pages, but it later appeared that EBay result pages now use java script to display

the set of results: therefore the tag set approach is irrelevant: since the data is not closed in html but rather the data is enclosed in java script code.

Therefore data extraction from the current EBay results pages will use a sequence of LANDMARKS that are sub-strings, as follows:-

1. Remove Preamble: Ignore all text until the sub-string 'dSI (' is found. This precedes the first data record. Each record starts with this sub-string.

2. Remove Post amble: The data section ends before the sub-string 'getResults (', so extract the text between the first 'dSI ('sub-string and the first 'getResults (' sub-string, before going on to step 3, to extract data from the records in this data section.

3. Each record contains a number of fields, but the interested items we want to extract from eBay are ITEM NUMBER, the item DESCRIPTION, the item PRICE and TIME LEFT. For each record we find them as follows:

Advance in the text to the next comma character. Do this fourteen (14) times, to ignore the first 14 data fields in the record. Text in the next comma is the ITEM NUMBER, so store it. Then text in the next comma is the ITEM DESCRIPTION string inside inverted commas (double quote chars).

With these two fields stored in a database we can retrieve the item's web page from EBay by using the URL: http://cgi.ebay.co.uk/ws/eBayISAPI.dll?ViewItem&item=5783642519

However, the item number (5783642519) at the end should be replaced as required.

4. Repeat steps 1 and 3 to extract data from all records on the results page.

5. To obtain the next page of results, or any query results page, use the URL produced by concatenating the following five strings:

(i) http://search.ebay.co.uk/

(ii) SEARCH WORDS with hyphens between them... e.g. Tanzania -Kilimanjaro

(iii) _W0QQfromZR40QQfrtsZ

(iv) P*50 where P is the page number required. 1, 2, 3, 4, etc.

(v) QQfsooZ1QQfsopZ1QQsojsZ1

For example, the URL:http://search.ebay.co.uk/tanzania-kilimanjaro_W0QQfromZR40QQfrtsZ0QQfsooZ1QQfsopZ1QQsojsZ1 will query eBay for the first page of results when searching for "Tanzania Kilimanjaro".

Observing the next page:

Any results page that has a next page of results will contain the sub-string 'Next' If we don't find this sub-string in a results page then that page is the last for the current query.

4.1.7 Java extractor

The interesting thing we find when we use a java program to query EBay is that the result page source code format is different from the source code we get for queries to a web browser window. So the extractor written by examining the source from the browser won't work.

In order to extract the data records, we write a java program to fetch the webpage from the EBay query universal resource locator (URL). The EBay web server responds by producing a third web page format in response to a java program.

The format of the page produced in response to java program from web server produces different format from the one produced by web browser window. So the algorithm given above needs modification. Remove preamble and post amble as per above algorithm. The following substring will be used as a LANDMARK for the extractor program.

1. Substring for description and item Number

<td class="ebcTtl">

2. Substring for Price

<td class="ebcPr">

3. Substring for number of binds.

<td class="ebcBid">

4. Substring for time left.

<td class="ebcTim">

5. substring for url

ebcTtl\"><a href="

For example, for extracting price, mark the end of the substring "<td class="ebcPr">. Find the position of the '>' character. This is the start of price, then advance to the next position where the characters '</' are encountered. Finally extract the string which is between these positions. Repeat similar operation for the rest of fields (description, URL, time left and item number). These sub-strings will be used as LANDMARKS for the extraction of meaningful data.

The Client interface component

The client interface component provides the users access to the application. It provides to user the following functions. Search interface where the user can provide the query to the database. The system executes the query and displays the result to the user.

Gives the user a form where he/she can register to the application. After registration the user gets the form to specify the number of queries that he /she want to monitor on EBay web site. The interface gives the client the option to change their queries.

The details for the component are given in chapter 6

The Alerter component

The component is responsible for checking updates from the database and mail management. The module constantly checks the updates every hour.

The details for the component are given in chapter 6

4.1.8 Persistence storage (MySQL)

The system is designed to use three tables. User, userProfile and webData.

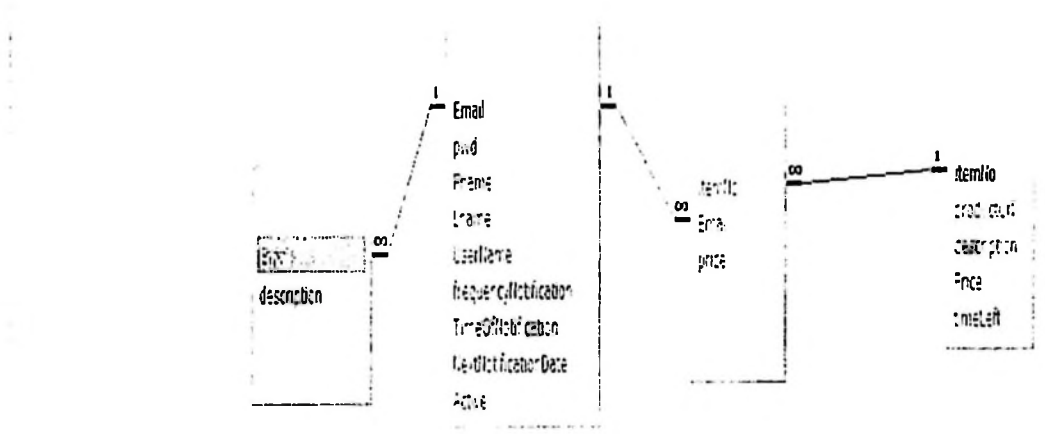


Figure 10: Database Schema.

Descriptions of tables used in the application.

User Profile table: The Table stores the items of the user's interests, which he/she want to search on EBay site. All of the keywords and phrases the user wants to extract are stored in the table. The table contains two fields, which are email and description.

Email is used to identify the user, while description stores the user's queries.

User table: The table store the user details, it has nine fields. (Email, pwd, fname, lname, username, frequencyNotification, timeofNotification, active). The table is used to store the user registration details. The user account is not active until the user replies by activating the mail by login when he/she receives mail from the system. The system requests the user to activate their accounts.

SavedQueries table: The table stores the items already notified to user. The table has three fields' itemNo, email, and price. The records stored in this table are used to compare with those from webData table. Since the data from webData table is updated regularly, it is easy to find the difference between the information stored. When the user is notified the table is also updated.

WebData table: The table contains the data extracted from EBay site according to queries which are stored in the user profile table. The table is updated regularly. Every day the information is extracted from EBay pages. The table has five fields (itemNo, productURL, description, price and time left.). The price and time left are expected to change, so when the extractor get the information from web pages it is looking for the updates in these fields.

4.2 SEMI AUTOMATIC WRAPPER

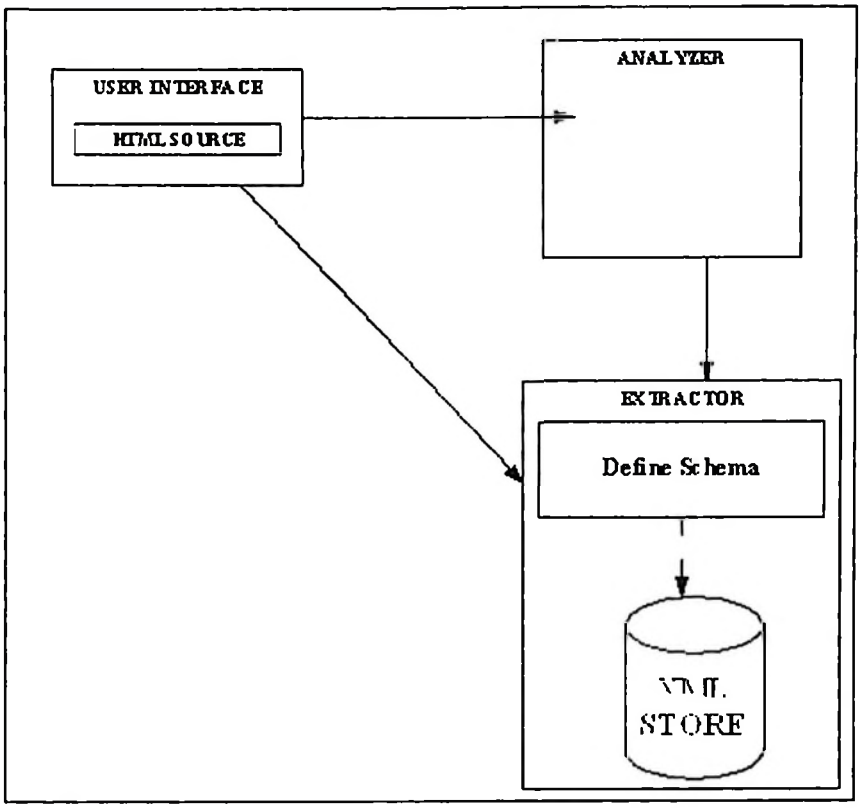


Figure 11: General Components for semi automatic wrapper.

The system has three components that are the user interface, Analyser and Extractor. General description for each component is described here; note the arrows show the flow of data from one component to another.

4.2.1 User interface component

The component is for providing the services to the end user. It is responsible for taking HTML source from the user and sends the HTML source to the ANALYZER. Another task performed by the component is to allow the user to define the Schema for the Extractor to use. The user is required to choose the cluster number as well as to defining the elements names (Fields); the extractor uses the defined field to label the data columns.

4.2.2 Analyser Component:

The analyser component is the important part of the wrapper, because it analyses the raw HTML source and produces all possible clusters in the document. The process of performing analysis has been explained in detail in the system analysis and the component follows a similar procedure. It produces the tagSet Progression grid in the form of two dimensional arrays of cluster numbers and TextString Number and sends the results of analysis to Extractor.

4.2.3 Extractor component

The Extractor component receives the analysed data from Analyser and extracts data according to the schema provided by the User interface component. In general the extractor is supposed to follow the fingerprint (TpGrid) produced by the analyser but the user may want to select few records from the fingerprint provided by the TpGrid. The algorithm used by the extractor to extract selected fields is explained here. First assume we have the TpGrid produced by the analyser as shown in figure 12

```
20. tagSet60: 60
21  tagSet61: 61,68,75,82,89,96,103,110,117,124,131,138,145,152,159,166,173,180,187,194
22. tagSet62: 62,69,76,83,90,97,104,111,118,125,132,139,146,153,160,167,174,181,188,195
23  tagSet63: 63,70,77,84,91,98,105,112,119,126,133,140,147,154,161,168,175,182,189,196
24. tagSet64: 64,71,78,85,92,99,106,113,120,127,134,141,148,155,162,169,176,183,190,197
25  tagSet65: 65,72,79,86,93,100,107,114,121,128,135,142,149,156,163,170,177,184,191,198
25. tagSet66: 66,73,80,87,94,101,108,115,122,129,136,143,150,157,164,171,178,185,192,199
27. tagSet67: 67,74,81,88,95,102,109,116,123,130,137,144,151,158,165,172,179,186,193
29. tagSet200: 200
```

Figure 12: TagSet Progression Grid.

The TpGrid is decomposed into array of text String values, each tagSet makes one array, and for example in Figure 12 will have seven arrays. Each TagSet makes its own array as shown in figure 13.

The arrays are used by the extractor to follow the textString, when the user defines the schema. For example the user can define TagSet 61 to take the logical name Title.

TagSet 62 to take logical name Description and finally define TagSet 63 to take name Price. The extractor takes these logical names with corresponding Tag set number and starts extracting values from the arrays in figure13. The algorithm takes the first element for each array to generate the first record. and then moves to the second element for each array again to generate the second record. The extractor will keep on moving across all arrays until it reaches the last record.

The Schema generated is in the format of XML file. The file which will be generated corresponding to the schema given above and the arrays in figure 13 is shown in figure 14:

```
   array 1
0  textString611
1  textString612
2  textString613
3  textString614
...............


   array 2
0  textString621
1  textString622
2  textString623
3  textString624
...............
    array 3
0  textString631
1  textString632
2  textString633
3  textString634
...............

    array 4
0  textString641
1  textString642
2  textString643
3  textString644
...............
    array 5
0  textString651
1  textString652
2  textString653
3  textString654
...............

    array 6
0  textString661
1  textString662
2  textString663
3  textString664
...............


    array 7
0  textString671
1  textString672
2  textString673
3  textString674
...............
```

Figure 13: Decomposed arrays from Tag set Progression Grid (TpGrid).

```
<webData>
        <web>
          <TITLE>textString611</TITLE>
          <DESCRIPTION>textString621</DESCRIPTION>
          <PRICE>textString631</PRICE>
          </web>
         <web>
          <TITLE>textString612</TITLE>
          <DESCRIPTION>textString622</DESCRIPTION>
          <PRICE>textString632</PRICE>
          </web>
        <web>
          <TITLE>textString613</TITLE>
          <DESCRIPTION>textString623</DESCRIPTION>
          <PRICE>textString633</PRICE>
          </web>
        <web>
          <TITLE>........</TITLE>
          <DESCRIPTION>..........</DESCRIPTION>
          <PRICE>...........</PRICE>
         </web>
 </webData>
```

Figure 14: XML document generated from the arrays of [TagSet, TextString] pair

4.3 Technology

The technologies used to implement both alerter system and semi automatic wrapper are:

4.3.1 Java

The Java language is a general purpose object oriented language which is highly portable. This means that an application written in java can run on different operating system platforms. Other powerful features of the language include: being strongly typed, has no pointers, no global variables, no global methods, efficient garbage collector etc. These characteristics make java a relatively secure language. The java language supports the development of large applications as well as small applications called applets. Applets are embedded into HTML pages.

4.3.2 LiveConnect

LiveConnect is the mechanism provided by Netscape to enable JavaScript to communicate with java classes. The mechanism allows JavaScript to use public variables, methods and properties from java class. With the use of live Connect; JavaScript program can communicate with applets by both reading and writing public fields and methods of the applet.

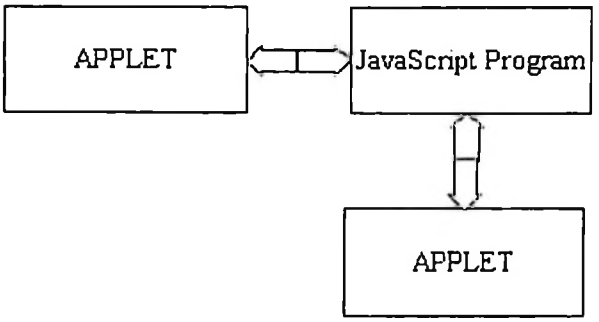


Figure 15: LiveConnect enables communication between applet and java Scripts program.

The reason why it was needed to use Live Connect was that I wanted to use Java and JavaScript in conjunction so as to benefit from the best features of both.

4.3.3 MySQL

MySQL has several useful features. Some of them are described here. The first remarkable feature is that MySQL database is designed with a portability capability. The portability is achieved by the use of three tools, which are GNU automaker (1.4), Autoconf (version 2.52) and libtool. The second feature is security; all password traffic is encrypted when connecting to a server. Finally, MySQL handles large data volume with highly concurrent access support.

4.3.4 JSP

Java server pages (JSP) is one of the leading technologies for developing dynamic web pages. JSP allow java class to be called in the jsp file. Also it uses the principle of "write once, run any where" that is to say that JSP pages are platform independent.

Environment for the system to operate

The semi automatic wrapper requires Java, JavaScript enabled browsers and Java run time Environment installed on the system. The alerter system needs web Server which supports java server pages (like Tomcat), MySQL database engine, mail server and Java run time Environment.

4.4 Summary

In this chapter we covered the design of both alerter and semi automatic web wrapper. The components involved in the architecture of the system were described. The algorithm used by the alerter as well as that used by the semi automatic wrapper to filter the extracted data were described. Finally the chapter discuss the technology used to implement the system.

5. IMPLEMENTATION

It is obvious that web pages may change in structure, data or both. So, for the ALERTER part of the project, the project assumes that the page structure is not changing and that only the data is changing. In the second part of the project which is semi automatic web wrapper production, changes in page structure are easily identified by the use of tag Set Progression Grid (tpGrid) [4] techniques. The numbers of clusters present in the HTML source code is identified before the data extraction is performed. This prevents the program from crashing as it becomes aware of the change in page structure and extracts data accordingly.

5.1 ALERTER SYSTEM

5.1.1 Wrapper

The wrapper component is responsible for fetching pages, downloading pages, extracting the useful data and saving them in the relational database. The wrapper component uses two classes called OnlineExtractor and AccessConnect. The OnlineExtractor is the class which is responsible for fetching, downloading and extracting data from pages. AccessConnect is responsible for database access (Connecting, adding, updating and closing). The relationship which exists between OnlineExtractror and AccessConnect is delegation. The OnlineExtractror delegates the database operations to AccessConnect. The methods which are delegated to AcessConnect are setConnection (), closeConnection () and saving Data to database.

The data types and operations which are involved in the wrapper component are discussed.

OnlineExtractor class

The purpose of the class is to fetch, download, and extract meaningful data from the raw html source pages. The attributes in OnlineExtractor class are:-

ItemNo: - It is of the type String; it is used to identify the product Number for the extracted data.

description: - it is of type String: It holds the description of the product extracted from the web pages.

ProductUrl : - it is of type String; it holds the Universal Resource Locator pointing to the product extracted from the page.

ItemPrice : - it is of type String; it holds the price of the product extracted.

TimeLeft: - it is of type String. it holds the remaining time for the auction to end for a particular product.

webData: - It is a collection of type Set. the variable hold Objects extracted from the web pages. The Objects in the webData are of type OnlineExtractor.

The operations present in the OnlineExtractor class are as described here under:-

getContents () :- The method gets contents of the page by taking universal resource locator (URL) as input and return collection of Objects of the Set type. The Objects in the set are of type OnlineExtractor.

checkingNextPage () :- The operation takes URL as input and return Boolean value. The methods used to check if there is a next page to download.

checkingRepeation () :- The method is used to remove redundant data from the collection and return collection without duplicates. It removes redundancy by making comparison between items Number of the downloaded products. The item number is unique for each product.

Run ():- it is responsible for. Scheduling the time for extraction and monitoring for any changes in the database. The thread is running after every twelve hours to download contents from EBay site.

The AccessConnect class has the following operations.

setConnection ():- It is used to make connection to the database.

closeConnection ():- It is used to close connection to the database.

savingData (Collection data):- The method takes collection as input and returns Boolean value. It is used to save data to the database. When the Extraction is complete the OnlineExtractor delegates the saving operation to AccessConnect to save extracted data to relational database.

updateExtractedData (Collection data):- The input is a Collection and the return type is a Collection of new Objects which were not updated.

display (String queries):- The method returns Collection of Objects from the database. The OnlineExtractor and AccessConnect classes are shown in figure 16.

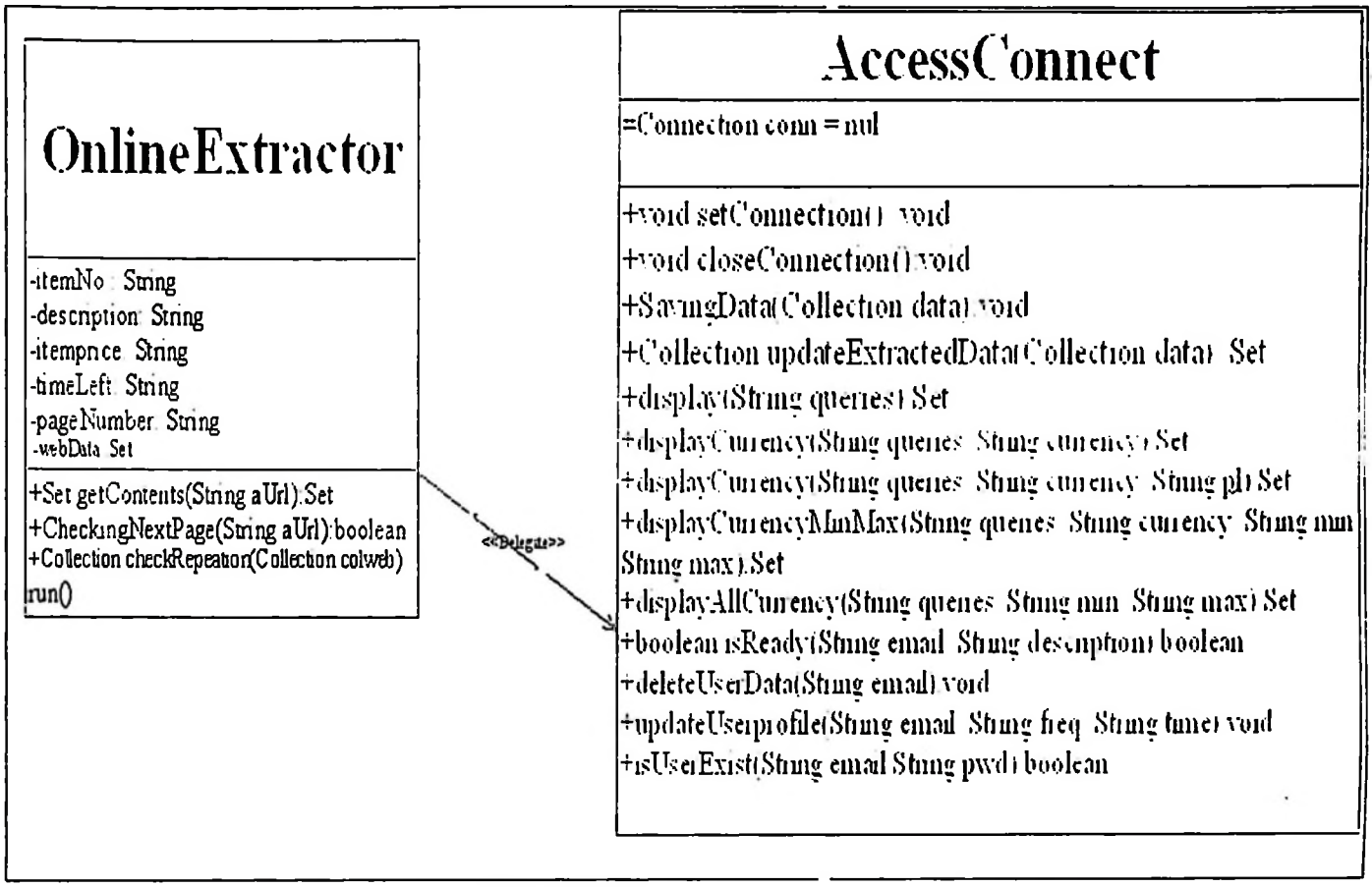


Figure 16: Class Diagram for Online Extractor and Access Connect.

5.1.2 Alerter component

The alerter component is responsible for monitoring the updates from the database and for issuing notifications. The alerter component is scheduled to hit database after every hour to see if there is any update in the items being monitored by particular users. The scheduling operation is done by the OnlineExtractor through method run () as a delegation operation.

The alerter component uses three classes; these classes are MailManager, AccessConnect and OnlineExtractor. The MailManager is the class which is responsible for mail composition and notification while AccessConnect is used for database operations like make connection and close connection.

The methods used by MailManager and AccessConnect are described below.

MailManager

mailSender () :-The method is responsible for actual action of sending mails.

dailyusers ():- The method returns a set of all active users who have requested alerts of products from EBay auction site. The method returns all users, who have requested notification within a range of one hour of the current time and the keywords saved for monitoring. The method uses system time to compare with the time user specifies to receive alerts.

mailComposer (String email, Collection col) :- The method takes email address and a set of data for a particular user and returns a String as a mail text. The collection contains all the keywords/items for a specific user.

mailOrganizer () :- The function which is performed by the method is to make sure the operations are done in a sequential order, starting from checking the existence of requester, then the composition of mail for specific requester and finally the dispatching of the composed mail. The OnlineExtractor delegate mail Operations to MailManager through mailOrganizer method. The MailManager, AccessConnect and OnlineExtractor classes are shown in figure 17.

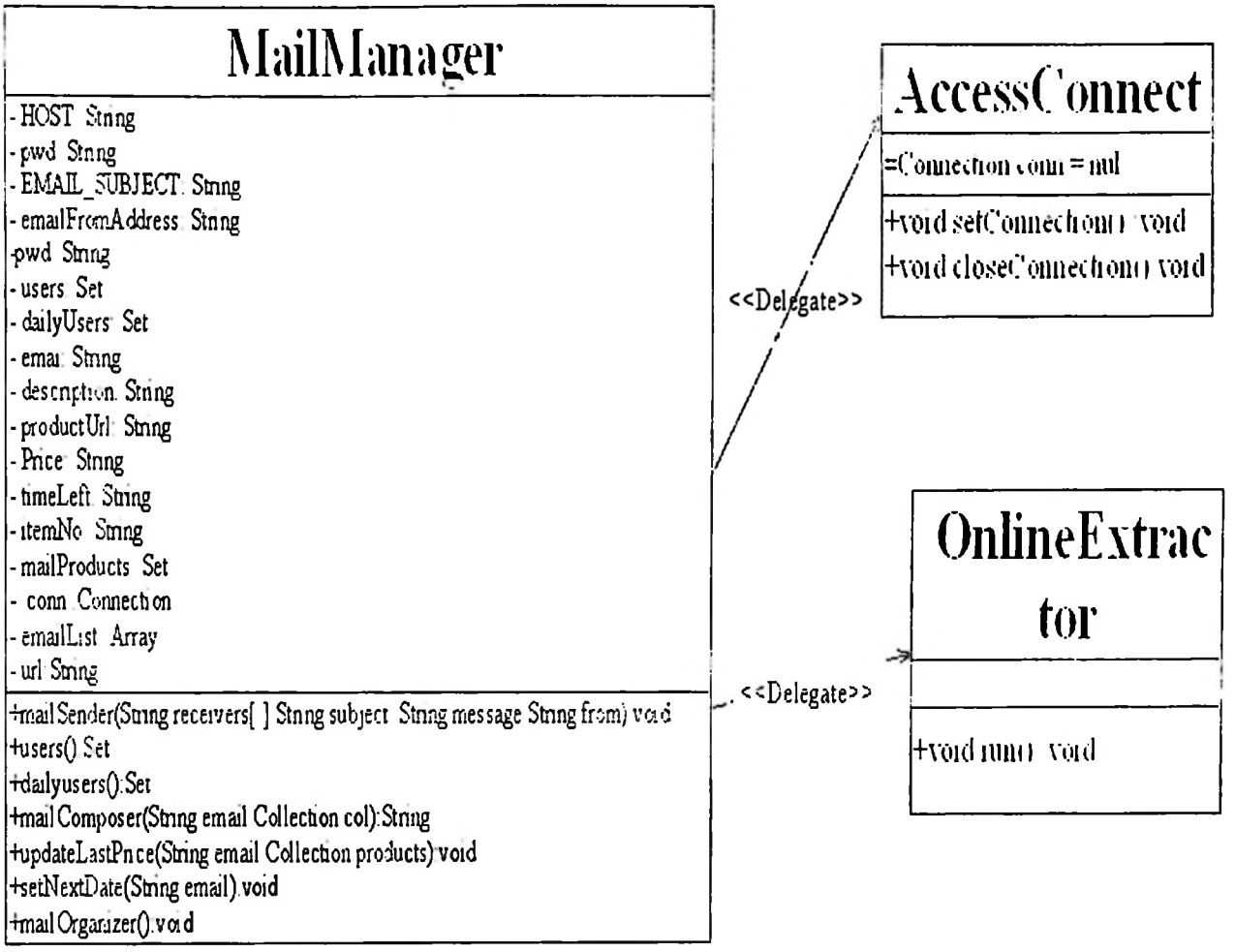


Figure 17: MailManager. AccessConnect and OnlineExtractor classless.

The MailManager class delegate the database operations to AccessConnect class and scheduling time to check updates from the site to OnlineExtractor class.

5.1.3 The Client interface component

The component is responsible for the interaction between user and the application. The component uses the web browser as a graphical interface. The user is supposed to type the universal resource locator (URL) for the application. It will display the front page for the application with a number of options for the user to use. Figure 18 shows a view of the front page for the application window.

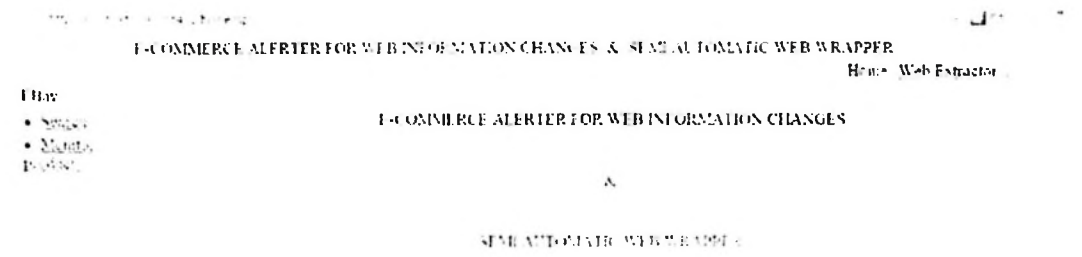


Figure 18: Front page window

The options provided are discussed here under.

1. Search option.

The search option is provided by the link button as shown in figure 18. When the search link is clicked the application display a screen as shown in figure 19. The screen provides text fields and one combo box for searching.

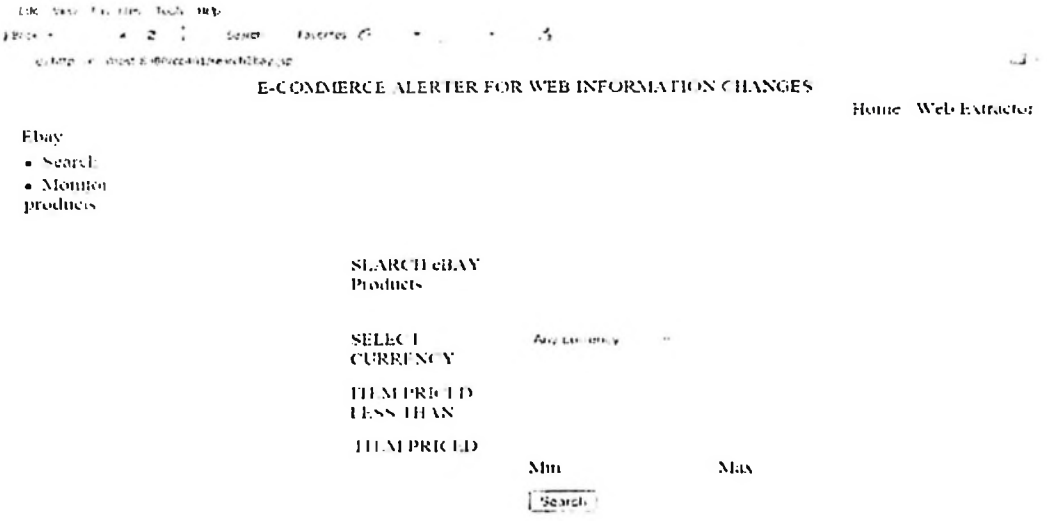


Figure 19: Search window.

For example when search is performed by using keyword 'java' and the currency is chosen as US dollars and the price between 15 and 20. the application shows a screen results as seen in figure 20.

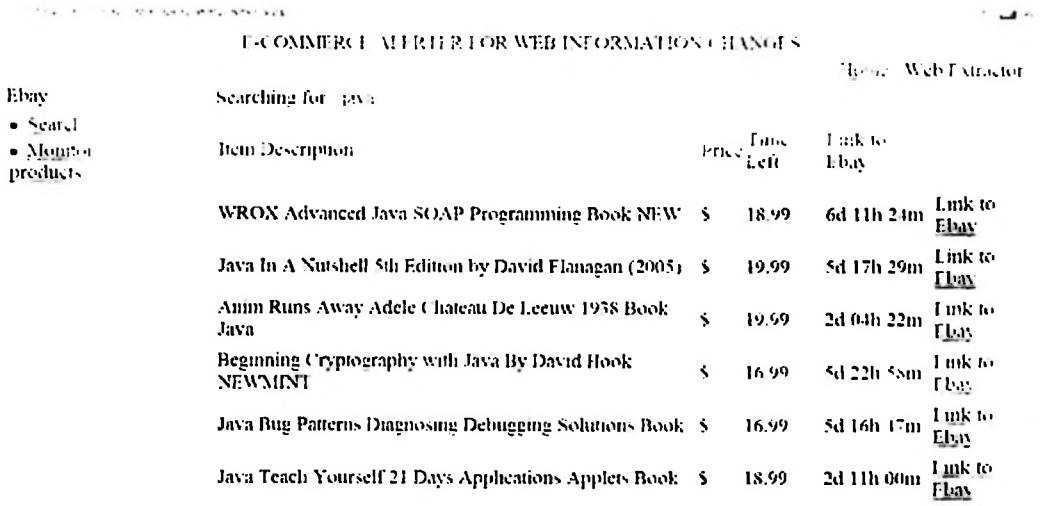


Figure 20: Search results.

2. The monitoring option requires a user to be registered. The process of registration starts as follows: The 'Monitor products' link is provided as shown in figure 18. The user has to click the monitor products button upon which the screen for registration will be displayed as shown in figure 21.

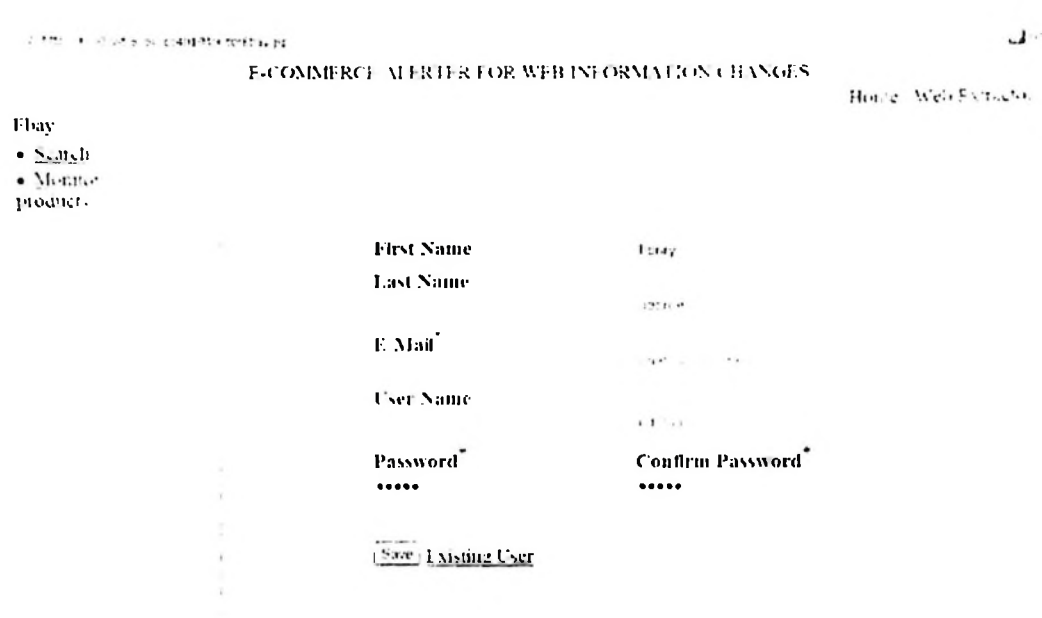


Figure 21: Login and registration screen.

For registration, the user has to provide the details required as shown in figure 21. The Save button is available for the user to save the details, as the button is clicked, the window will open where he/she can write a number of queries for the system to monitor. On clicking on save button the monitoring screen is shown as in Figure 22.

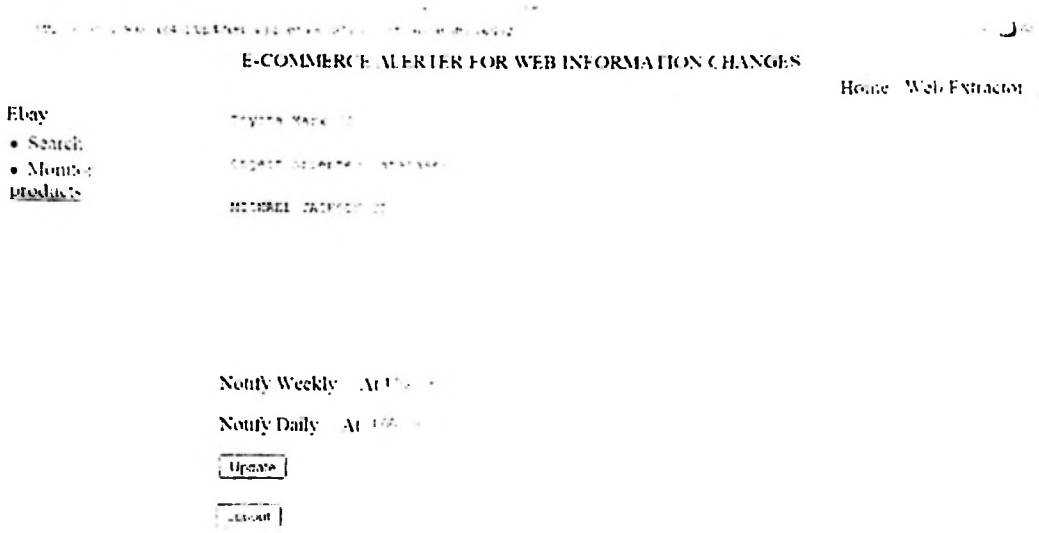


Figure 22: Monitoring Screen.

The monitoring screen provides text Field area where the number of queries can be written or changed. After writing queries for monitoring the user is supposed to click update and then logout. The text Fields will increment automatically as the number of queries increases. The user will not receive any notification messages until he/she accepts to receive them. When the user is registered for the first time the mail is generated to ask for the permission to send notification messages from the application. The user is supposed to click on the link provided and then to login for the confirmation to receive notifications.

For the user to make changes on their request/data they want to monitor, they need to Login by clicking on the Existing User link. The link directs the user to the login Screen. The existing user link is shown in figure 21. When the existing user Link is clicked the login page will be shown as in figure 23.

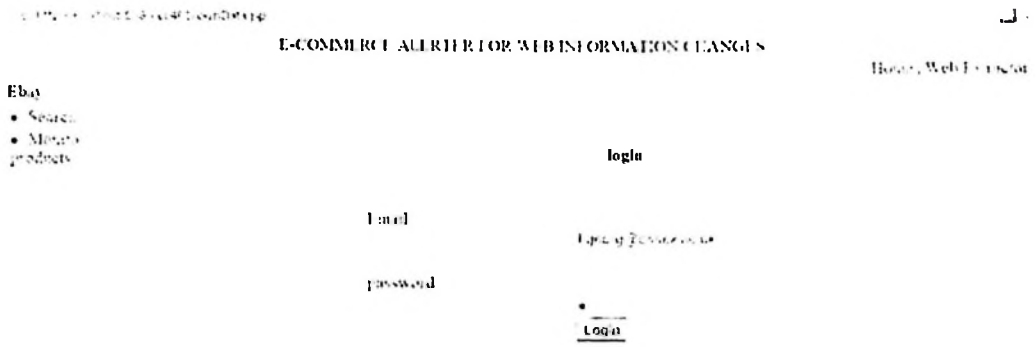


Figure 23: Login

The page displays a screen with two text fields. For the user to log into the system, the email address and password are required. The successful login will display a list of previously saved queries as shown in figure 5. The queries may be altered or removed.

5.2 SEMI AUTOMATIC WRAPPER GENERATION

The Semi automatic wrapper is responsible for analysing HTML source code, Defining the schema of the extracted data and producing structured data in the format of XML. The component needs human involvement in the process of defining schema. The component has three subparts: the user interface, TpGridApplet class and TestApplet class. The function performed by user interface is to receive HTML source code and to define data schema. The TestApplet receives HTML source and analyzes it, the TpGridApplet receives the outcome of analysis from TestApplet and extracts data according to the schema defined in the user interface. The class diagram for the classes used is shown in figure 24.

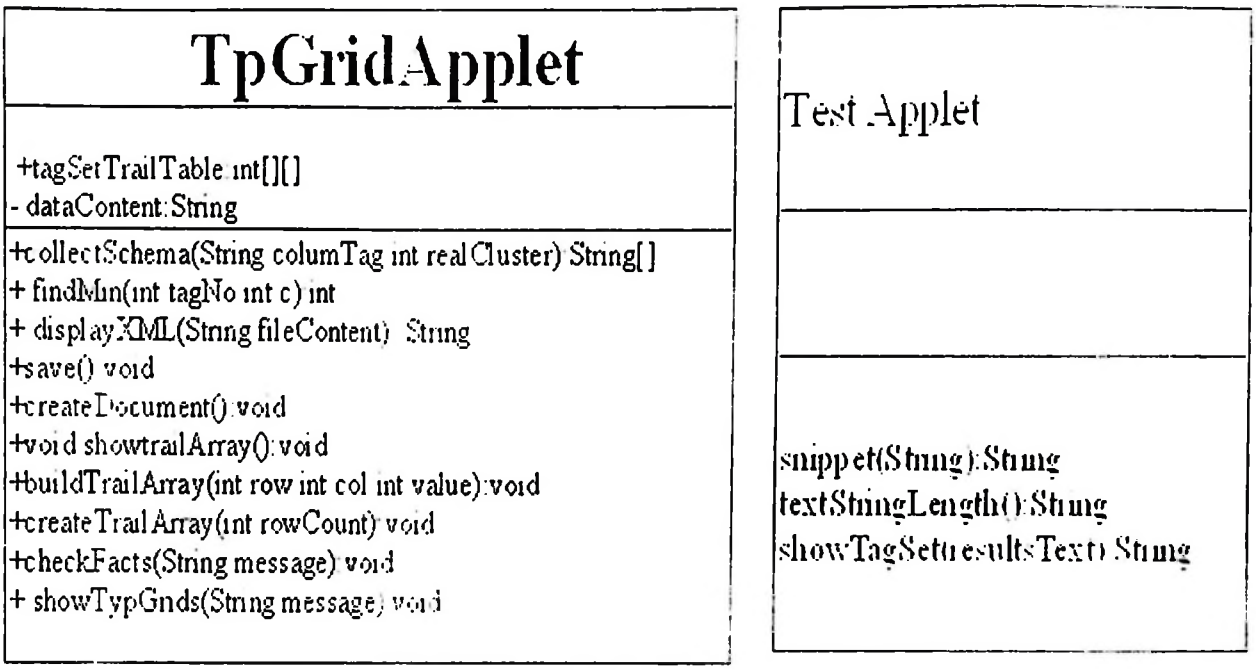


Figure 24: Class Diagram for TpGridApplet and TestApplet.

In general, the class TestApplet is used to analyse HTML source. It produces the tagSet progression grid. The TpGridApplet uses the fingerprint of the analyzed results from TestApplet to operate.

5.2.1 TpGridApplet class

The main purpose of the class is to extract data from HTML source code with the help of human involvement. The Class receives cluster number and data schema from a person as well as TpGrid map from TestApplet. The implementation of the algorithm used to pick data items from the tag set progression grid produced by TestApplet is implemented in the method called collectSchema.

Let's discuss one particular method called collectSchema. It is the method which implements the algorithm discussed in chapter five for TpGrid decomposition. The method takes TagSet Number and cluster number, and then it fetches all text String belonging to a given TagSet number in a specified cluster number. The source code for

method is shown in figure 25: the method collectSchema is used to filter data according to the user specification, meaning that a record in the web page may have many fields, but the user may require only few on them or all of the fields.

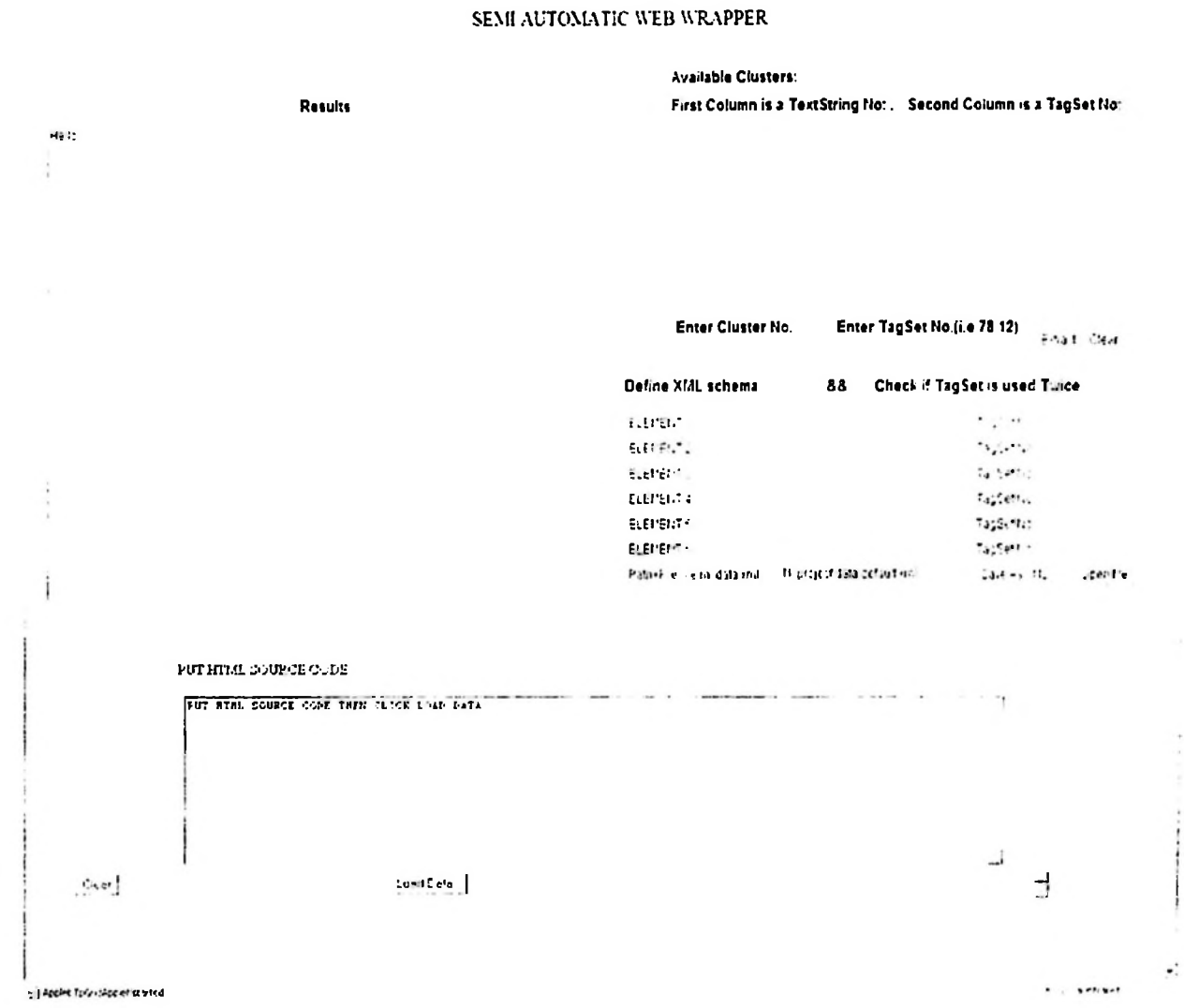
```
/**
Method receives TagSet number and cluster number
It
It returns array of textstring for a specific TagSet Number in given cluster
*/
public String[] collectSchema(String columTag, int realCluster) {
        String columG[] = new String[10000];
        isEndofDoc = false, vatcout = 0,realClusterFound = false,
        String dataModify = "", int tagStringNo = 0;
        boolean first = false, int tmpCluster,
        first = false;
        int i = 0,
        while (i < textString.length && isEndofDoc == false) {
            if (i < tagSetTrailTable.length) {
                tmpCluster = tagSetTrailTable[i][1];
                tagStringNo = tagSetTrailTable[i][0],
                Integer itTagNo = new Integer(tagStringNo);
                try {
                    if (realCluster == tmpCluster && columTag equalsIgnoreCase(itTagNo toString())) {
                            if (first == false) {
                                int tmpMin = findMin(tagStringNo, tmpCluster),
                                int tmpMax = findMax(tagStringNo, tmpCluster);
                                if (tmpMax == tagStringNo) {
                                    // The loop will continue to extract data from the
                                    dataModify = textString[tmpMin - 1];
                                    vatcout=0,
                                    columG[vatcout] = dataModify,
                                    vatcout++;
                                    dataModify = textString[tagStringNo],
                                    columG[vatcout] = dataModify;
                                     first = true;
                                   } else {
                                    vatcout=0;
                                    dataModify = textString[tagStringNo];
                                    columG[vatcout] = dataModify;
                                    first = true;
                                   }
                                realClusterFound = true;
                                first = true,

                            } else {
                                // The loop will continue to extract data from
                                // the second record up to the last in the cluster
                                vatcout++;
                                dataModify = textString[i];
                                columG[vatcout] = dataModify,
                                realClusterFound = true;
                                }                                }

                } catch (Exception e) {
                    generalResults.append( Error );
                }
            } else {
                isEndofDoc = true,
            }
            i++,
        }
        if (realClusterFound == false) {
            generalResults.append( No record found  );
        } else {
            realClusterFound = true,
        }
        return columG;
    }
```

Figure 25: collectSchema method used to collect data from the TpGrid.

After method collectSchema is executed, we will have a one array of text String for each tagSet number in a given cluster. The arrays will be used to generate schema according to the user specifications. The algorithm of making schema will start as follows: start with the first array, take the first element and put it into XML file as the first field, move to the next array, take the first value again and put it in the XML document as the second field. Similarly by accessing the first elements of all of the remaining arrays and inserting them into the XML file will give us the first record. A similar procedure would be followed for the rest of the records, whereby the second elements of all the arrays will be picked for the constitution of the second record, the third element for the third record and so on. The generated XML document will look like figure 24.

5.2.2 User interface

The user interface is a browser based graphical interface with a number of text area boxes, command buttons and text fields for user to communicate with the wrapper. Figure 26 shows a view of the wrapper.



Figurer 26: Semi automatic wrapper

The HTML source code is required in the text area "PUT HTML CODE", then click load Data to send the html source to TestApplet. TestApplet returns the number of clusters associated with Tag Set and text String numbers. In the text field "Enter Cluster No" and "TagSetNo" is for user to view the data available in each cluster. The area "Define xml Schema" is for user to define logical names for the tagSet number. User is supposed to choose cluster number, provide the meaningful name in the "ELEMENTx. x represent that any logical name is acceptable provided that does not start with a number" with corresponding target number then provide the file name with xml extension. The operations explained above will look like figure 27.

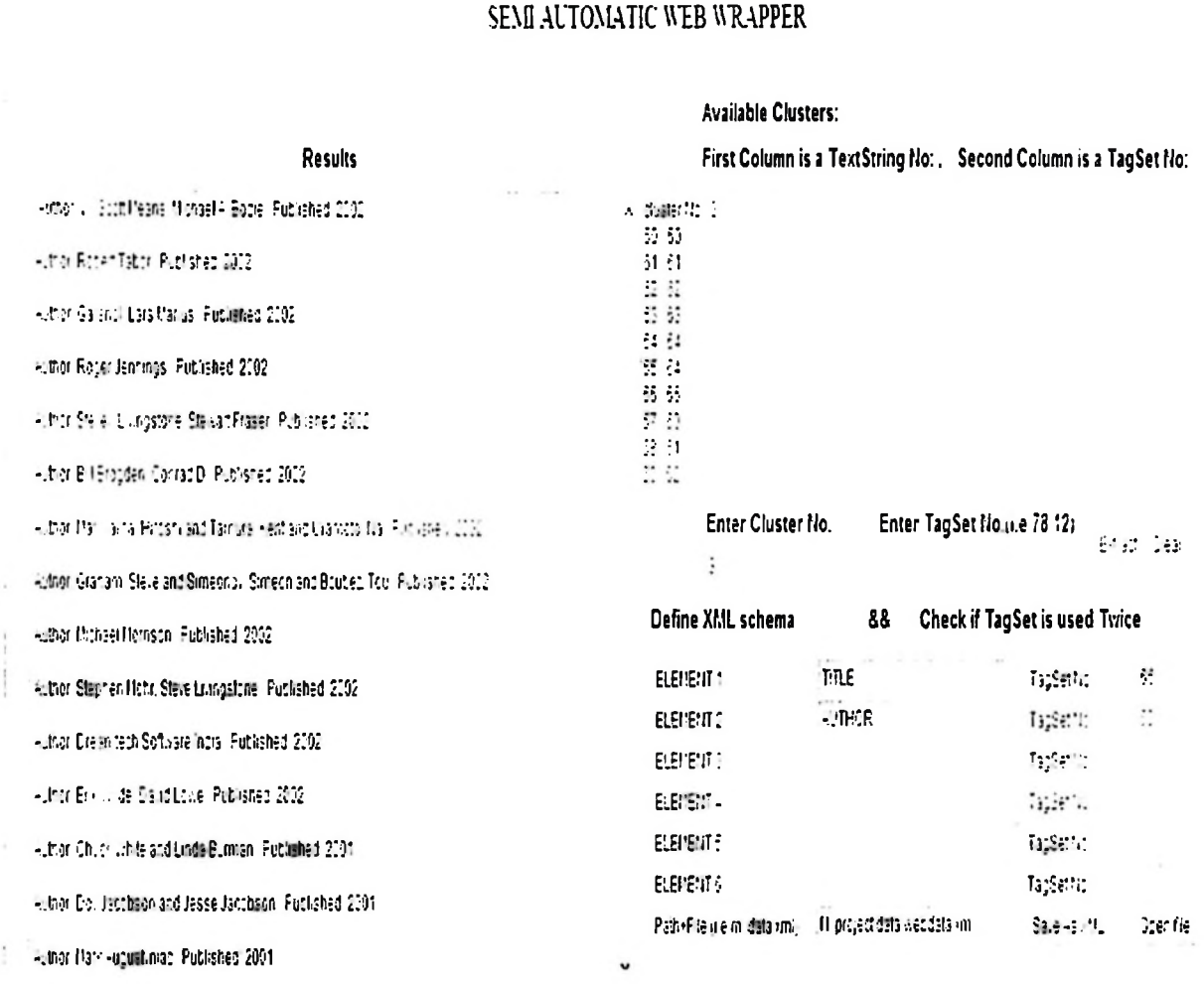


Figure 27 Defining XML schemas and generation of XML document.

The XML document which was generated from Figure 27 is shown in figure28: the html source has been wrapped to XML according to the user's requirements. As we can see, the XML document has two elements, TITLE and AUTHOR. The XML document file could be easily used by another application.

```
<?xml version="1.0" encoding="UTF-8" ?>
<WebData>
- <web>
    <TITLE>The Book of SAX: The Simple API for XML</TITLE>
    <AUTHOR>Author: W. Scott Means, Michael A. Bodie Published: 2002</AUTHOR>
  </web>
- <web>
    <TITLE>Microsoft .NET XML Web Services</TITLE>
    <AUTHOR>Author: Robert Tabor Published: 2002</AUTHOR>
  </web>
- <web>
    <TITLE>Definitive XML Application Development</TITLE>
    <AUTHOR>Author: Garshol, Lars Marius Published: 2002</AUTHOR>
  </web>
- <web>
    <TITLE>Visual Basic.NET XML Web Services Developers Guide</TITLE>
    <AUTHOR>Author: Roger Jennings Published: 2002</AUTHOR>
  </web>
- <web>
    <TITLE>Beginning VB.NET XML: Essential XML Skills for VB.NET Programmers</TITLE>
    <AUTHOR>Author: Steven Livingstone, Stewart Fraser Published: 2002</AUTHOR>
  </web>
- <web>
    <TITLE>Cocoon 2 Programming: Web Publishing with XML and Java</TITLE>
    <AUTHOR>Author: Bill Brogden, Conrad D Published: 2002</AUTHOR>
  </web>
- <web>
    <TITLE>XML and Java(TM): Developing Web Applications</TITLE>
    <AUTHOR>Author: Maruyama, Hiroshi and Tamura, Kent and Uramoto, Na Published: 2002</AUTHOR>
  </web>
- <web>
    <TITLE>Building Web Services with Java: Making Sense of XML, SOAP, WSDL and UDDI</TITLE>
    <AUTHOR>Author: Graham, Steve and Simeonov, Simeon and Boubez, Tou Published: 2002</AUTHOR>
  </web>
- <web>
```

Figure 28: The XML document generate by using semi automatic web wrapper.

5.3 Summary

We have covered the implementation of both alerter system and semi automatic web wrapper. The classes involved and their relationship between them. The algorithm which is used to filter data from tag set progression grid was explained in the method called collectSchema. Finally the user interfaces for both system where shown.

6. TESTING

The purpose of testing is to find the difference between the observed behaviour and the expected behaviour of the system [22]. So the main objective is to detect errors/faults that may exist in the system.

From the engineering point of view, testing should be carried out in three stages.

1. Unit testing: This is the process of checking individual components of the system. The components are tested using stubs and test drivers [22].
2. Integration testing: It is the process of testing integrated components. Several components are assembled and then tested. It detects faults that have not been detected during unit testing [22].
3. System testing: It is the process of checking the entire system to ensure that it complies with functional and non functional requirement [22].

6.1 Alerter system

Unit testing: The system has been decomposed into two sub systems. The first component is the web wrapper which consists of the downloader, data extractor and saving of the extracted data and the second component is the alerter component.

Web wrapper: The test conducted for web wrapper follows black box testing. The wrapper was given a query "database books" to extract from www.eBay.com. The wrapper managed to extract all data items found as the results of the query on EBay auction. This was done by comparing the extracted data items and those found on the web browser page for similar query.

The second component is alerter, the component was tested by using black box testing, the component was given email address and mail message to send notification. The component manages to do the work of dispatching the mails successfully.

Integration testing: The wrapper, data store and alerter components were integrated for testing. Three keywords were stored in the database for the wrapper to use to query information from www.eBay.com. As a result the wrapper managed to use the stored keywords and extract data accordingly. The alerter uses the same keywords to search from the downloaded data and then compose and send electronic mail according to the requester.

System testing: The whole system starting with java server pages, wrapper and alerter component were integrated and then tested. The user was able to register to the system, perform operations for adding, removing and changing the keywords for wrapper and alerter to use. When the user registered to the system with keywords to monitor, the system managed to use the keywords to fetch data items and notify changes accordingly.

6.2 Semi automatic web wrapper

To measure the success of our semi automatic web wrapper we compute recall and precision rations for the extracted data for seven web sites. The experimental metrics are defined as follows,

- Recall (R) is a proportion of the correctly extracted data items of the all data items that should be extracted.

$$\mathrm{Re}call(R) = \frac{\#CE}{\#PC}$$

- Precision is a proportion of the correctly extracted data items of all the data that has been extracted.

$$\mathrm{Pr}ecision(P) = \frac{\#CE}{\#R}$$

Where: -

#CE=Number of correct records Extracted.

#PC =Number of possible correct records

#R= Number retrieved.

We present the results from seven web sites which were chosen for the testing of the semi automatic web wrapper. Five pages were taken from each site and the summary of extracted data is tabulated as shown in figure 29.

URL	Search Result Records	Records Extracted	Correct Records Extracted	Recall	Precision
http://www.shoe-shop.com/bin/venda	80	60	60	75%	100%
http://www.halfpricecomputerbooks.com	100	95	95	95%	100%
http://a1books.com	75	74	69	92%	93.24%
http://www.mamma.com	105	99	99	94%	100%
http://www.dogpile.com	80	78	74	92.5%	94.8%
http://campus.acm.org	100	100	100	100%	100%
http://webcrawler.com/	100	100	100	100%	100%

Figure 29: Summary of data extracted showing recall and precision.

6.3 Experimental discussion on the Extracted data

The results of semi automatic web wrapper are displayed in figure 29. We are explaining why wrapper manages to achieve precision and Recall of 100% for some sites while the percentage is far lower for others.

Starting with **http://www.halfpricecomputerbooks.com** the wrapper managed to extract 95 records from the cluster but the number of records present in the source was 100. Upon taking a closer look at the structure of the tag set progression grid as shown in appendix B we can find that for the first three pages, there exists a notable difference between the first page and the rest of the pages. The first page has a structure which looks different from the rest of the pages. The structure is not the same because in the first page it has image in the first record while other records have no images. The wrapper failed to extract the five records because of the additional attributes (image). As we can see on the appendix B, the tagSet 89 in the first page is used as unique tagset while on the rest of the pages it is represented by tagset 61. The presence of image in the first page causes tagset 89,117, 152,138, 173 to be displaced from their positions. Therefore in the first page TagSet 89 represents the separate attribute of the record while in page 1 and page 2 it is represented by tagset 61.

The wrapper managed to achieve a Recall of 92% and Precision of 93.24 % of the data extracted from **http://albooks.com.** It fails to reach 100% because of the missing attributes in some of the records. If we take a look at the appearance of Progression Grid (TpGrid) as shown in the appendix C, we can figure out the missing attributes on some records. The appendix C on page one shows that tag set 52 and 102 are displaced from their actual position and similarly for page three, tag set 77 and 142 are displaced. For example in page three the sequence expected for the tag set were (74-75-76-77-78) and (139-140-141-*142*-143) but the tag set 77 and 142 were missing from their positions. Also if we take a look at the records displayed on the browser we can see that some records have missing attributes for authors. The missing attributes cause the wrapper to fail to work as desired.

In the case of **http://campus.acm.org** and **http://webcrawler.com**, the wrapper managed to achieve the Precision and Recall of 100% because the appearance of the Progression Grid (TpGrid) shows that the records were complete, no nested iterations and there were no missing attributes.

In general the tests done for semi automatic web wrapper are encouraging as seen in table 29: the recall and precision reach up to 100% for well formed sites (web pages with records without nested iterations and missing attributes). The experiments shown in figure 29 with high recall and precision prove to be a great success for data extraction, although problems such as the handling of missing values, nested iterations and multi-valued attributes still persist. The part of data extracted from http://www.halfpricecomputerbooks.com is shown as appendix A.

6.4 Summary

The methods used for testing alerter and semiautomatic web wrapper were discussed. We use black box test for alerter system due to the nature of the project being of research based and dynamic nature of web pages. The approach used to test semi automatic web wrapper is by using sample pages. The results were tabulated and discussed, due to irregularity nature of HTML source, some web pages were not extracted as was expected but those with regular, with no nested iteration and no missing values were successfully extracted.

7. CONCLUSION

7.1 Achievements

In the achievements of the project we need to discuss factors, which indicate the success or failure of the project. These indicators are objectives of doing the project, final product meeting the functional and non-functional requirements and successful project management.

Objectives of the project

The objectives of doing the project were clearly stated in the project proposal and the area which is in particular demand apropos the proposed system. However, as it was a research intensive project, with no clearly formulated or structured design of the final product from the outset—what I wanted to achieve, the way of going about doing it and the limitations of implementation, became clear to me as the project went along. Needless to say, the horizons of my vision broadened as I gained more knowledge and expertise and I gradually began to gain an in-depth understanding of the vast and intricate field of data extraction. So eventually the project was more like a journey of understanding and discovery, rather than towards a well-known destination.

Meeting functional and non-functional requirements

The project managed to fulfil most of the functional and non-functional requirements as were stated in the system analysis.

For the case of Electronic commerce alerter system, the project has been able to fulfil the following functionalities.

1. The system extracts data from web pages (www.eBay.com) and stores the structured data to a relational database (MySQL Database).
2. The system provides search functionality from the local database.

3. The system provides the web page monitoring functionality and reports the changes which appear on the extracted data.
4. The system provides the automatic updating in the Relational database by extracting new updates on the web pages.

For the case of semi automatic web wrapper, the project was able to achieve the following functionalities.

1. The wrapper accepts HTML source code as a source of information.

2. The system allows user to define the schema and subsequently extract data according to the defined schema.

3. The wrapper allows the extracted data to be saved in XML format.

Project management:

The project was completed within the time specified and only the available resources were used. The scheduling of activities and rectifying problems that crept up due to a particular activity lagging behind was a big hurdle I faced. Planning of the project was a real challenge, especially for a novice and that too for a challenging area like data extraction.

The above indicators show that the project was successful because the project was well managed and the software produced meets most of the functional requirements.

7.2 Problems encountered

In general we are saying the project was successful but we did encounter some problems. The problems encountered are stated and the solutions of the problems are given here.

Problems on dynamic structure page changes

The process of pages analysis, using sub-strings as LANDMARKS to extract data, can have several unpredictable consequences when the page structure changes. It has been

observed that the EBay site changes frequently. As a result the extractor must be rewritten otherwise it won't work. The structure of the part of the page is given to show how the structure changes may affect the data extraction.

```
<tr class="single">
<td class="ebLeft"> </td>
<td class="ebcPic"><a href="http://cgi.ebay.com/ws/eBayISAPI.dll?ViewItem&amp;category=38179&amp;item=
<td class="ebcTtl"><a href="http://cgi.ebay.com/ws/eBayISAPI.dll?ViewItem&amp;category=38179&amp;item=
<td class="ebcPr"><span class="bold">$1.99</span><br /></td>
<td class="ebcBid"><img border="0" title="Buy It Now" alt="Buy It Now" src="http://pics.ebaystatic.com
<td class="ebcStr"><a href="http://stores.ebay.com/id=26163341?sspagename=L2">Our Books and Thing<wbr/>
<td class="ebRight"> </td></tr>

<tr class="ebHlOdd single">
<td class="ebLeft"> </td>
<td class="ebcPic"><a href="http://cgi.ebay.com/ws/eBayISAPI.dll?ViewItem&amp;category=38179&amp;item=
<tr class="single">
<td class="ebLeft"> </td>
<td class="ebcPic"><a href="http://cgi.ebay.com/ws/eBayISAPI.dll?ViewItem&amp;category=38179&amp;item=
<td class="ebcTtl"><a href="http://cgi.ebay.com/ws/eBayISAPI.dll?ViewItem&amp;category=38179&amp;item=
<td class="ebcPr"><span class="bold">$2.99</span><br /></td>
<td class="ebcBid"><img border="0" title="Buy It Now" alt="Buy It Now" src="http://pics.ebaystatic.com
<td class="ebcStr"><a href="http://stores.ebay.com/id=16269144?sspagename=L2">CoffeeRoastersClub<wbr/>
<td class="ebRight"> </td>
</tr>

<tr class="ebHlOdd single">
<td class="ebLeft"> </td>
<td class="ebcPic"><a href="http://cgi.ebay.com/ws/eBayISAPI.dll?ViewItem&amp;category=38179&amp;item=
<td class="ebcTtl"><a href="http://cgi.ebay.com/ws/eBayISAPI.dll?ViewItem&amp;category=38179&amp;item=
<td class="ebcPr"><span class="bold">$2.99</span><br /></td>
<td class="ebcBid"><img border="0" title="Buy It Now" alt="Buy It Now" src="http://pics.ebaystatic.com
<td class="ebcStr"><a href="http://stores.ebay.com/id=21217?sspagename=L2">cbCOFFEE<wbr/></a></td>
<td class="ebRight"> </td>
</tr>
```

FIGURE 26: Before page changes

```
<tr class="ebHlOdd single">
<td class="ebLeft"> </td>
<td class="ebcCpr"><input type=            name=           value=              onclick=
<td class="ebcPic"><div class=             style=                                ><a href=
<td class="ebcTtl"><a href="http:    .ebay.com      -Five-                          
<td class="ebcPpl"> <img src="http:  pics.ebaystatic.com/aw pics/paypal logo_paypalPtbuyerProtection
<td class="ebcPr"><span class="bold">$99.00</span><br /></td>
<td class="ebcBid">1</td>
<td class="ebcTim">5h 24m</td>
<td class="ebRight"> </td></tr>


<tr class="single">
<td class="ebLeft"> </td>
<td class="ebcCpr"><input type="checkbox" name="coitem" value="     13682631" onclick=
<td class="ebcPic"><a href="http:  cgi.ebay.com Relational Database Design Clearly Explained
<td class="ebcPpl"> <img src="http:   
<td class="ebcPr"><span class="    ">$1.00</span><br /></td>
<td class="ebcBid">-</td>
<td class="ebcTim">16h 23m</td>
<td class="ebRight"> </td></tr>

<tr class="ebHlOdd single">
<td class="ebLeft"> </td>
<td class="ebcCpr"><input type="checkbox" name="coitem" value="7174022187" onclick=
<td class="ebcPpl"> <img src="http:  pics.ebaystatic.com/aw pics/paypal logo_paypalPtbuyerPr
<td class="ebcPr"><span class="bold">$39.99</span><br /></td>
<td class="ebcBid">1</td>
<td class="ebcTim">1d 09h 50m</td><td class="ebRight"> </td></tr>
<tr class="single">
```

FIGURE 27: After page changes

The figure 26 shows the part of the page when the page was analysed for the first time and the figure 27 shows the structure when the page was changed.

For example for the Extraction of Item number, the extractor uses the following algorithm. Initially the extractor looks for the substring '<td class="ebcTtl">', then finds the word 'item=', and extracts ten characters followed by "item=".

In the second analysis, when the page changed, the sub string "item=" is not present anymore, so the extractor will not work.

The page analysis was done again and in the newer implementation the extractor looks for the sub string "coitem value =", which is followed by ten digits which is the item number. The data extraction using occurrences of sub-string, therefore changes in web pages may lead to the tedious job of extractor maintenance. The structural changes on the web page require a new analysis, and re-writing of the web page data extractor.

However, the problem wherein a new wrapper has to be written every time the page structure changes can be solved by the use of tag set progression grid.

When the page is analysed for the first time the Tag set Progression Grid is obtained. This can be used as a fingerprint for the next extraction. So the next time data needs to be extracted, the page will be analysed and compared against the previous Tag set Progression Grid. If the comparison reveals any differences, it indicates that the page structure has changed since the last time the page was analysed.

Subsequently, necessary actions are required to be taken to remedy the situation and overcome the adverse effects on wrapper generation due to the changes: and before any users recognize the failure of the system: therefore trigger will be set to notify the system administrator to warn about the failure of wrapper.

For the case of re-writing a wrapper we have developed a tool which helps the developer to extract the items of interest in the XML format rather than HTML. However, the afore-mentioned process of using this wrapper to detect changes in page structure and notify the system administrator were not implemented due to time constraints.

Page format

The problem of HTML format produced by the web Browser and Java program was one of the problems I faced. In the beginning when I was doing page analysis I was using HTML page source produced by the Internet Explorer Browser to find the landmarks that could be used to locate data items. But when I use the java program to download HTML source, the server response to the program by producing HTML page which has different format from that produced by the browser. Therefore the landmarks obtained by analysing HTML page using source produced by the browser were not used to locate data items for the implementation of E-commerce alerter system because the format is different to those obtained by java program. To cope with the problem I decided to do analysis again using the page downloaded by the java program to get the landmarks.

Presence of nested data structure iteration and missing values

The tag set progression grip for page representation works fine for the pages without nested structures and missing values. As we can see in table 29 the precision and recall is not 100% because the clusters contain values which are nested and some of which are missing. For example http://www.halfpricecomputerbooks.com produces a cluster with one record missing a field. As a result the wrapper failed to extract five records correctly out of 100 records.

Changes made on the project proposal

Initially the project was focussing on the www.ebay.com as a case study for data extraction and eventually development of E-Commerce alerter system using Tag set Progression Grid. But when the analysis was done on the HTML page from the www.ebay.com it was realised that data items where enclosed in the JavaScript, which means the Tag set approach was irrelevant. Then the project was adjusted to allow page analysis by using sub-strings as landmarks for the extractor to use, also to add the second part of the project, which is semi automatic web wrapper.

Page format

The problem of HTML format
of the problems I faced. In
HTML page source produced
could be used to locate
source, the server respon
format from that produ
analysing HTML page
items for the implem
different to those obtai
analysis again using the

Presence of nested data

The tag set progressi
nested structures and
not 100% because the
missing. For example
one record missing a field as a result the
out of 100 records.

Changes made on the project proposal

Initially the projec
extraction and ev
Progression Grid
www.ebay.com
means the Tag se
analysis by using
part of the projec

8. BIBLIOGRAPHY

[1] Doorenbos .R, Etzioni.O, Weld.D (1997) 'A Scalable Comparision-Shopping Agent for the World Wide Web'. Proceedings of the first international conference on Autonomous agents, pages 39-48. New York, NY, USA . 1997

[2] J. Robinson 'Analysing Web Pages for Automatic Wrapper Production in Data Extraction'. http://cscourse.essex.ac.uk/course/cc433/publications/wrappers.pdf . 2004 Last time accessed: 10th September, 2005

[3] David W. Embley.and Cui Tao.(2005) ' Automating the Extraction of Data from HTML Tables with Unknown Structure', pages 3-28, Elsevier Science Publishers B. V. Amsterdam, The Netherlands , July 2005.

[4] Robinson Jerome (2004), ' Data Extraction from Web Data Sources'. Proc WBC'04, 4th International Workshop on Web Based Collaboration, 2004.

[5] Robinson Jerome(2004), 'Data Extraction from Web Pages containing Query Results' http://cscourse.essex.ac.uk/course/cc433/publications/magazine.pdf . 2004

Last time accessed: 10th September, 2005

[6] Calton Pu, Ling Liu, Wei Tang (2000), 'Detecting and delivery information changes on the web'. Proceedings of the ninth international conference on Information and knowledge management, pages 512-519, ACM Press New York, NY, USA, 2000

[7] Shian-Hua Lin, and Jan-Ming Ho(2002), 'Discovering Informative Content Blocks from Web Documents'. Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining Pages, pages 588-593, New York, USA, 2002

[8] David Buttler,Daniel Rocco,Ling Liu (2004), 'Efficient Web Change Monitoring with Page Digest'. Proceedings of the 13th international World Wide Web conference on Alternate track papers & posters, Pages: 476 - 477 , New York, USA, May 2004

[9] J. Hammer, H. Garcia-Molina, J. Cho, R. Aranha, and A. Crespo (1997), Extracting Semi structured Information from the Web. In *Proc. Workshop on Management of Semistructured Data*, Tucson, Arizona , 16 May, 1997

[10] Muslea I., Minton.S. and Knoblock.C. (2000). 'Hierarchical wrapper induction for semistructured information sources'. Journal of AutonomousAgents & Multi-Agent Systems , pages 93-114. 2000

http://www.isi.edu/~muslea/PS/jaamas-2k.pdf

[11] Eikvil L.,'Information Extraction from World Wide Web - A Survey', Report No. 945. ISBN 82-539-0429-0. July 1999

[12] Hsu.C (1998). 'Initial Results on Wrapping Semistructured Web Pages with Finite-State Transducers and Contextual Rules'. AAAI-98 Workshop on AI and Information Integration. 1998.

[13] Soderland S.(1999). 'Learning Information Extraction Rules for Semistructured and Free Text. Machine Learning', 1999

http://citeseer.ifi.unizh.ch/cache/papers/cs/1943/http:zSzzSzwww.cs.washington.eduzSzhomeszSzsoderlanzSzWHISK.pdf/soderland99learning.pdf

Last time accessed: 5th September. 2005

[14] Sergio F., Filippo F. and Elio M.(2001). 'monitoring web information changes'. Proceedings. International Conference, Page 421 – 425. 2001

[15] Jerome Robinson(2004). 'Providing Robust Access to Data in Web Pages'.Technical report. University of Essex. March 2004

[16] C a l i f. M .Raymond J (1999). 'Relational Learning of Pattern-Match Rules for Information Extraction'. Proceedings of the sixteenth national conference on Artificial intelligence and the eleventh Innovative applications of artificial intelligence conference innovative applications of artificial intelligence. Pages: 328 - 334 . American Association for Artificial Intelligence Menlo Park. CA. USA . 1999

[17] Robert Baumgartner. Oliver Fr¨olich. Georg Gottlob. Patrick Harz. Marcus Herzog. Peter Lehmann. 'Web Data Extraction for Business Intelligence: the Lixto Approach'.

http://www.dbai.tuwien.ac.at/proj/lixto/WebBI.pdf

Last time accessed: 10th september. 2005

[18] *Bing Tan, Schubert Foo, Siu Cheung* Hui(2001). 'Web information monitoring: an analysis of Web page updates'. Online Information Review. Volume 25. Number 1. Pages: 6-19. February 2001.

http://www.emeraldinsight.com/Insight/viewPDF.jsp?Filename=html/Output/Published/EmeraldFullTextArticle/Pdf/2640250101.pdf

Last time accessed: 10th September, 2005

[19] Kushmerick Nicholas, Weld D, and Doorenbos R (2001). "Wrapper Induction for Information Extraction". Proc. 1997 International Joint Conference on Artificial Intelligence (1997) , Pages729-735. February 2001.

[20] Muslea I., Minton S., Knoblock A.(1998)'STALKER: Learning extraction rules for semistructured, Web-based information sources'. In Proceedings of AAAI-98 Workshop on AI and Information Integration. Technical Report WS-98-01. AAAI Press, Menlo Park, CA . 1998

[21] http://www.oasis-open.org/cover/rss.html(RDF Rich Site Summary)

Last time accessed: 10th September, 2005

[22] http://sh628.essex.ac.uk/course/cc453/slides74-113.pdf
Last time accessed: 10th September, 2005

[23] http://rss.softwaregarden.com/aboutrss.html

Last time accessed: 10th September, 2005

9. APPENDIX


```
<?xml version="1.0" encoding="UTF-8" ?>
- <WebData>
  - <web>
      <LISTPRICE>81.94</LISTPRICE>
      <authors>Author: Roger E. Sanders, Janet Perna Published: 2000</authors>
      <Price>40.97 Our Price:</Price>
      <Title>DB2 Universal Database Application Programming Interface (Api) Developers Guide</Title>
      <Extra3>40.97</Extra3>
    </web>
  - <web>
      <LISTPRICE>59.98</LISTPRICE>
      <authors>Author: George Baklarz, Bill Wong, Jonathan Cook Published: 2000</authors>
      <Price>29.99 Our Price:</Price>
      <Title>DB2 Universal Database v7.1 for UNIX, Linux, Windows and OS/2 Database Administration Certification Guide
        (4th Edition)</Title>
      <Extra3>29.99</Extra3>
    </web>
  - <web>
      <LISTPRICE>29.99</LISTPRICE>
      <authors>Author: Paul Collins Published: 2000</authors>
      <Price>23.99 Our Price:</Price>
      <Title>Exam Cram Oracle8 Dba: Database Administration</Title>
      <Extra3>6.00</Extra3>
    </web>
  - <web>
      <LISTPRICE>59.98</LISTPRICE>
      <authors>Author: Roger Jennings Published: 1999</authors>
      <Price>29.99 Our Price:</Price>
      <Title>Database Developers Guide with Visual Basic 6</Title>
      <Extra3>29.99</Extra3>
    </web>
  - <web>
      <LISTPRICE>99.98</LISTPRICE>
      <authors>Author: Microsoft Corporation Published: 1999</authors>
      <Price>49.99 Our Price:</Price>
      <Title>Microsoft Sql Server 7.0 Database Implementation Online Training Kit</Title>
      <Extra3>49.99</Extra3>
```


Appendix A: Extracted data from http://www.halfpricecomputerbooks.com.

Appearance Progression Grid (tpGrid) of the first three pages for "http://www.halfpricecomputerbooks.com".

```
Page one

tagSet60: 60
tagSet61: 61,68,75,82,   96,103,110,    124,131,    145,    159,166,    180,187,194,201
tagSet62: 62,69,76,83,90,97,104,111,118,125,132,139,146,153,160,167,174,181,188,195,202
tagSet63: 63,70,77,84,91,98,105,112,119,126,133,140,147,154,161,168,175,182,189,196,203
tagSet64: 64,71,78,85,92,99,106,113,120,127,134,141,148,155,162,169,176,183,190,197,204
tagSet65: 65,72,79,86,93,100,107,114,121,128,135,142,149,156,163,170,177,184,191,198,205
tagSet65: 66,73,80,87,94,101,108,115,122,129,136,143,150,157,164,171,178,185,192,199,206
tagSet67: 67,74,81,88,95,102,109,116,123,130,137,144,151,158,165,172,179,186,193,200
tagSet89: 89,117,138,173
tagSet152: 152
tagSet207: 207

Page two

tagSet59: 59
tagSet60: 60,67,74,81,88,95,102,109,116 123,130,137,144,151,158,165,172,179,186,193
tagSet61: 61,68,75,82,89,96,103,110,117,124,131,138,145,152,159,166,173,180,187,194
tagSet62: 62,69,76,83,90,97,104,111,118,125,132,139,146,153,160,167,174,181,188,195
tagSet63: 63,70,77,84,91,98,105,112,119,126,133,140,147,154,161,168,175,182,189,196
tagSet64: 64,71,78,85,92,99,106,113,120,127,134,141,148,155,162,169,176,183,190,197
tagSet64: 65,72,79,86,93,100,107,114,121,128,135,142,149,156,163,170,177,184,191,198
tagSet66: 66,73,80,87,94,101,108,115,122,129,136,143,150,157,164,171,178,185,192
tagSet199: 199

Page three

tagSet59: 59
tagSet60: 60,74 67,81,88,95,102,109,116,123,130,137,144,151,158,165,172,179,186,193
tagSet61: 61,68,75,82,89,96,103,110,117,124,131,138,145,152,159,166,173,180,187,194
tagSet62: 62,69,76,83,90,97,104,111,118,125,132,139,146,153,160,167,174,181,188,195
tagSet63: 63,70,77,84,91,98,105,112,119,126,133,140,147,154,161,168,175,182,189,196
tagSet64: 64,71,78,85,92,99,106,113,120,127,134,141,148,155,162,169,176,183,190,197
tagSet64: 65,72,79,86,93,100,107,114,121,128,135,142,149,156,163,170,177,184,191,198
tagSet66: 66,73,80,87,94,101,108,115,122,129,136,143,150,157,164,171,178,185,192
tagSet199: 199
```

Appendix B: The Appearance Progression Grid (tpGrid) from http://www.halfpricecomputerbooks.com/.

The Appearance Progression Grid [illegible] three pages from http://www.a1books.com:8080

Page One

tagSet33: 33
tagSet34: 34,39,44,49,54,59,64,69,74,79,84,89,94,99,104,109,114,119,124,129,133,138,143,148,153
tagSet35: 35,40,45,50,55,60,65,70,75,80,85,90,95,100,105,110,115,120,125,130,134,139,144,149,154
tagSet36: 36,41,46,51,56,61,66,71,76,81,86,91,96,101,106,111,116,121,126,131,135,140,145,150,155
tagSet36: 37,42,46, 57,62,67,72,77,82,87,92,97 ,107,112,117,122,127, 136,141,146,151,156
tagSet38: 38,43,48,53,58,63,68,73,78,83,88,93,98,103,108,113,118,123,128,132,137,142,147,152
tagSet52: 52,102
tagSet157: 157

Page Two

tagSet33: 33
tagSet34: 34,39,44,49,54,59,64,69,74,79,84,89,94,99,104,109,114,119,124,129,134,139,144,149,154
tagSet35: 35,40,45,50,55,60,65,70,75,80,85,90,95,100,105,110,115,120,125,130,135,140,145,150,155
tagSet36: 36,41,46,51,56,61,66,71,76,81,86,91,96,101,106,111,116,121,126,131,136,141,146,151,156
tagSet36: 37,42,47,52,57,62,67,72,77,82,87,92,97,102,107,112,117,122,127,132,137,142,147,152,157
tagSet38: 38,43,48,53,58,63,68,73,78,83,88,93,98,103,108,113,118,123,128,133,138,143,148,153
tagSet158: 158
tagSet160: 160

Page Three

tagSet33: 33
tagSet34: 34,39,44,49,54,59,64,69,74,79,84,89,94,99,104,109,114,119,124,129,134,139,144,149,154
tagSet35: 35,40,45,50,55,60,65,70,75,80,85,90,95,100,105,110,115,120,125,130,135,140,145,150,155
tagSet36: 36,41,46,51,56,61,66,71,76,81,86,91,96,101,106,111,116,121,126,131,136,141,146,151,156
tagSet36: 37,42,47,52,57,62,67,72, 82,87,92,97,102,107,112,117,122,127,132,137, 147,152,157
tagSet38: 38,43,48,53,58,63,68,73,78,83,88,93,98,103,108,113,118,123,128,133,138,143,148,153
tagSet77: 77,142
tagSet158: 158
tagSet160: 160

Appendix C: The Appearance Progression Grid [illegible]