A TAXONOMY AND ANALYSIS OF WEB WRAPPING TECHNOLOGIES

Shin Wee Chuang

Working Paper CISL# 2004-08

June 2004

Composite Information Systems Laboratory (CISL)
Sloan School of Management, Room E53-320
Massachusetts Institute of Technology
Cambridge, MA 02142

A TAXONOMY AND ANALYSIS OF WEB WRAPPING TECHNOLOGIES

by

Shin Wee Chuang

Submitted to the Engineering Systems Division and the Department of Civil and Environmental Engineering on May 7th, 2004,
in Partial Fulfillment of the Requirements for the Degrees of
Master of Science in Technology and Policy
And
Master of Science in Civil and Environmental Engineering

ABSTRACT

Web wrapping technologies were developed in the 90s in the middle of the dot com boom to facilitate the extraction of web data. In recent years, the underlying architecture of web wrapping technologies is also been used for other applications such as information integration between legacy systems in large enterprises.

Despite the relatively widespread use of this technology, there is currently no uniform way of characterizing web wrapping toolkits, unlike say, a digital camera which can be described in terms of the size of its sensor or storage capacity. The focus of this thesis therefore is to develop a taxonomy or classification scheme that can be used to effectively describe a web wrapping toolkit in terms of its retrieval, extraction and conversion features. For this purpose, some 20 toolkits are studied and of which, verification tests were performed on 9 of these toolkits where evaluation copies are available.

The last part of the thesis discusses two policy Acts that are closely related to data extraction. They are the EU Database Directive and the HR3261 Database and Collection of Information Misappropriation Act. A comparative analysis between the two Acts was performed and their respective implications on the database producing industry were examined.

Thesis Supervisor: Stuart Madnick
John Norris Maguire Professor of Information Technology and Professor of Engineering Systems

Thesis Reader: John Williams
Professor of Civil and Environmental Engineering and Engineering Systems

Chapter 1: Introduction

The Internet has come of age since the older days of the ARPANET. Today, the Internet is ubiquitous and is woven into the fabric of our everyday lives, from checking emails to booking an exotic holiday. We use the Internet to connect to people from all over the world, for both work and leisure. We use the Internet to search for research materials that we need for our school projects. We use the Internet to search for the ideal holiday getaway. We use the Internet to search for the latest news about sports, entertainment, book reviews, and just about everything. In fact, the Internet has become the 411 of the modern era that we now live in.

At the same time that the Internet is revolutionizing our personal lives, it is also slowly creeping into the work place as well. Companies now rely on the Internet to gather information, to perform market analysis, to source for the cheaper materials and to connect offices in disparate locations. However, since the Internet is designed primarily for human consumption, such activities have to be performed manually. This is a costly exercise for such a mundane effort. Fortunately, the emergence of web wrapping technologies takes away the need for such repetitive actions to be performed manually. Computer programs are written that automatically extracts the required data from the Internet, and produce an output format that are deemed fit by the end-users. No longer is manual browsing of the Internet necessary in order to perform competitors' analysis, and even the filling up of online forms can be automated. Web wrapping toolkits have, in a way, connected the disparate of information in the world-wide-web into a consolidated, giant "virtual database".

Not all web wrapping toolkits are capable of performing all the required tasks, however. There is an abundance of web wrapping toolkits out there that cater to different needs. Some specialize in converting an input format to a different output format; others are better at handling more complex web pages such as the ones with embedded javascripts. Still, others are simply excellent acting as an integration platform to connect disparate data sources, from both internal file systems and external public web pages. In this thesis, a new taxonomy that is aimed at better classifying web wrapping toolkits will be described so that readers can have a better understanding of this technology.

1.1 Outline of the Thesis

The rest of this thesis is organized as follows. Chapter 2 contains a brief technical introduction of web wrapping technologies and its related applications. The differences between a web wrapping toolkit and a web wrapper are first discussed. This is then followed by a literature review of the past methodologies presented by other researchers.

Chapter 3 discusses the taxonomy of web wrapping toolkits. Here, a "bottom up" approach is introduced in the construction of web wrapping taxonomy. Using a 2-tier system, the individual attributes used in the classification are first explained before the 2-tier taxonomy, which shows the comparative features between all the 20 toolkits surveyed, is revealed.

Chapter 4 deals with the analysis of the web wrapping toolkits that are included in this survey. A brief explanation of the features supported by each toolkit is provided, and as far as possible, a discussion on the architectural framework of the underlying technology is also provided.

Chapter 5 is a case study that demonstrates how the 2-tier taxonomy is used in practice. The aim of this chapter is to show how the taxonomy can be used to address a hypothetical question: "Which toolkit is the best substitute to Cameleon?"

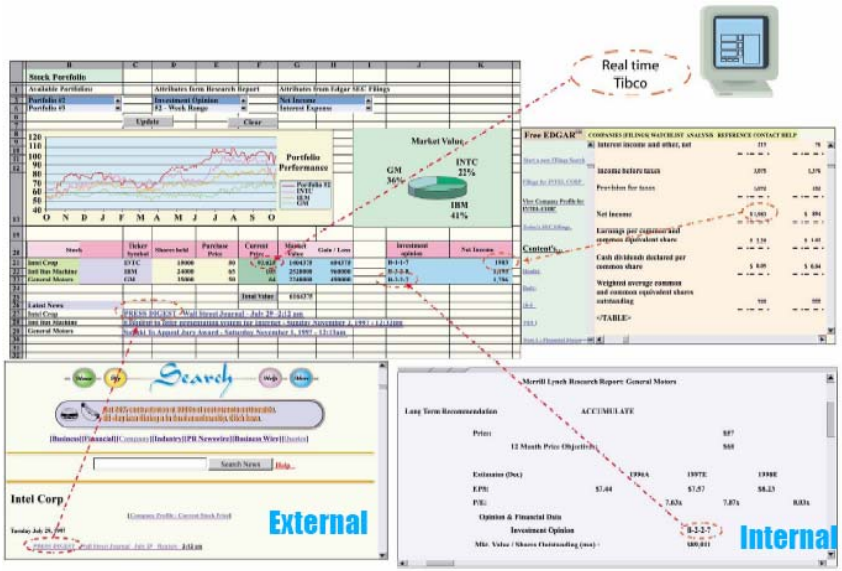
Chapter 6 is a policy discussion on the EU Database Directive and the HR3261 Database and Collection of Information Misappropriation Act. An introduction to the EU directive as well as the HR3261 bill is first described, then compared and analyzed. This is followed by a discussion on the implication of the 2 Acts on the database industry and the greater research community before a list of recommendations is presented at the end.

Chapter 7 is the conclusion of this thesis where a review of the web wrapping technologies and related policy implications is briefly described.

Chapter 2: Introduction to Web Wrapping Technologies

The phenomenal growth of the Internet in recent years has altered the way businesses are conducted throughout corporate America and elsewhere in the world. No longer is a presence in the world-wide-web considered a luxurious enclave exclusive to the Silicon Valley high-tech start-ups. Increasingly, more and more traditional brick-and-mortals companies are rushing into publishing their products and services online. Every day, millions of web pages are uploaded and updated with new information, and gigabytes of data are being thrown into the ever-expanding world-wide-web every second. As the amount of online data balloons and the use of the Internet increases, many organizations begin to rely on the web as an information source to conduct strategic activities such as competitors' analysis, market intelligence and data mining, where analysts are deployed to visit and revisit a certain number of web sites constantly to gather new information. While the use of the web for data mining-related activities represents a more efficient shift away from the traditional labor-intensive field-works, a better and more automated solution to manual web browsing is clearly needed as more and more web sites are used as information sources.

It is on this pretext that web wrapping technologies were developed. Software developers first wrote simple programs to extract segments of data from certain web data sources. These programs, known as web wrappers, automatically searched the Internet to look for relevant information queried by the users. Over time, more and more sophisticated features and functionalities are added to create more robust web data extraction products. In fact, the last few years witness a gradual shift in which web wrapping software are increasingly marketed and used as integration solutions. The wrappers generated from these products are not only able to extract information from static HTML web pages, but are also capable of distilling data from many internal file systems, such as an Excel file or an email attachment. Figure 1 below provides a graphical illustration of what a modern web wrapping toolkit is capable of doing:



Source: MIT Open Courseware

Figure 1: Web Wrapping Tool that extracts internal and external data sources

2.1 Web Wrapper and its Application

The emergence of aggregators further accelerated the development of web wrapping technologies. An aggregator is an online service provider that takes a query from a user, deciphers the query, sends out web wrappers to various web sites to gather information, and then returns the "aggregated" data back to the user. A familiar example of an aggregator is mysimon.com, which sends out web wrappers to query the databases of online merchants, using a set of input parameters defined by the user. For example, if a user wants to buy an iPod MP3 player, he only needs to type the word "ipod" in the "search" textbox. Mysimon.com will then display a list of online merchants that sell iPod, along with the price of the product, taxes, shipping charges and other information. Figure 2 below shows a partial list of online merchants selling Apple iPod (40GB).

Figure 2: The screen shot of an aggregator

In the remainder of this chapter, a more formal and detailed explanation about web wrapper and its accompanying technologies will be provided. Then, the various past approaches that have been taken to categorize or define web wrappers will be discussed.

2.2 Web Wrapper vs. Web Wrapping Toolkits

A Wrapper, by definition, is a specialized computer program that is capable of extracting data from web pages and transforming them into structure formats. In general, the way it works is by accepting queries from the users, goes to the web page(s) where the information resides (either through a direct URL or a set of URLs), extracts the requested information (according to a set of pre-defined extraction rules) and returns the result in a certain format. A web wrapping toolkit, on the other hand, is a piece of software that facilitates the generation of web wrappers through its engine that usually specifies the techniques of extraction, extraction rules, accessible data types, data transformation algorithm and so on.

As a general rule of thumb, a robust web wrapper is characterized by three main functions: retrieval, extraction and conversion. To satisfy the retrieval requirement, a web wrapper must at least be able of accessing a targeted web page through HTTP methods such as GET and POST. Advanced wrappers are usually capable of handling more sophisticated tasks such as refreshes, redirections, authentications and interpreting script languages, amongst others. Beyond retrieval, a web wrapper must also be capable of identifying a certain targeted segment of a web page for extraction. Finally, the conversion mechanism of the wrapper saves the extracted data into a structured format, which are then usually exported to other application for further processing.

In the early days of web wrapping, a wrapper needs to be written manually by an expert programmer using one of the general-purpose programming languages such as Perl or JAVA. This is however not a trivial task as the programmer has to write a different program for each website, making this manual approach both time-consuming and error-prone. These hand-written wrappers also cannot adapt to any changes to the web site layout and structure, and have to be modified for each new change, resulting in significant maintenance costs. As the number of sources of interest grows, hand-written wrappers become infeasible. It is therefore imperative that, aided by advances in artificial intelligence and machine learning, new wrappers that can be semi-automatically or automatically generated started to appear over the last few years.

Wrappers that are semi-automatically generated differ from their manual counterparts in that the user usually makes use of a graphical interface to “teach” the wrapper-generating toolkit what information to extract, and from where. This series of step-by-step demonstration is then converted into codes that form the basis of a semi-automatically generated wrapper. It is called semi-automatic because of the need for the user to show the system how to extract the relevant information, including entering some keywords such as passwords, where applicable. This approach is far less tedious and error-prone than the manual approach, but the wrappers still need to be updated for each time there is a structural change to the web sites. It is also not possible for the system to induce the structure of similar sites, so demonstrations have to be performed for each new web site that needs to be extracted.

For automatically generated wrappers, machine learning algorithms such as inductive learning are used. Typically, a set of delimiter-based extraction rules are generated by feeding the toolkit with training examples. However, this type of wrappers generally returns varying result depending on the quality of the given training examples and the complexity of the web sites that are wrapped. For very complex web sites, the automatic approach might prove to be more

expensive than even the manual approach since a large number of training examples have to be supplied and then supervised.

2.3 Literature Review of past characterization of web wrapping toolkits

There is some literature devoted to the characterization of web data extraction tools, or toolkits. For example, Kuhlins et al simply divided the toolkits into 2 general categories: commercial and non-commerical[1]. Laender et al, on the other hand, categorized these toolkits into ways in which the wrappers are generated, such as the wrapper development languages, HTML-awareness, natural language processing, induction, modeling and ontology[2]. In the same vein, Firat developed a 3-dimensional matrix in classifying academic toolkits, based primarily on how mechanism behind wrappers generation[3]. Having already discussed the 3 ways in which a wrapper can be generated, in the following section, the other two categorization proposed by Firat for the categorization of academic toolkits will be discussed. Then, the taxonomy that Laender proposed will be surveyed.

2.4 Firat's Matrix

Aykut Firat proposed a 3-dimensional matrix for categorizing web wrapping toolkits. Designed primarily for the academic toolkits, he concluded that a web wrapping toolkit could be described in three different ways: 1) how it generates wrappers; 2) whether it treats web pages as document tree or as a data stream; and 3) whether the specification files (or extraction rules) are declarative or not. In the following section, we shall discuss about the methodologies used by him.

2.4.1 Document Tree vs. Data Stream

A certain number of web sites today make use of the hierarchical relations between various HTML elements in crafting out their web pages. Taking advantage of this tree structure, some toolkits are designed to produce wrappers that parse these web pages, treating the web pages as a document tree (abbreviated to WAD by Firat), usually using the Document Object Model (DOM) as a basis for their extraction rules. The DOM is basically "a platform- and language-neutral interface that will allow program and scripts to dynamically access and update the content, structure and style of documents."[4] Wrappers built around this structure have several advantages, including ease of use in writing extraction rules by utilizing the HTML tag hierarchy. However, such an approach is likely to increase the maintenance cost in the long run because of the need to constantly update the wrappers with each changing HTML version and the need to introduce error recovery mechanism for HTML pages that do not conform to this tree structure.

At the other end of the spectrum, other toolkits ignore the HTML tag hierarchy and treat web pages as a data stream (abbreviated to WAS by Firat), using regular expressions to define the extraction rules in most instances. The immediate advantage of adopting this approach is the expanded ability of these web wrappers to not only extract data from traditional HTML-based web pages, but also from other text documents including email messages, XML pages and plain

text. The drawback for treating web documents as a data stream is that the extraction rules of these wrappers are comparatively harder to write, especially if the wrappers are to be manually generated.

2.4.2 Declarative

In general, a toolkit is considered to be declarative if there is "a clean separation of extraction rules from the computational behavior of the wrapping engine"[3]. In the same vein, non-declarative toolkits usually produce wrappers in which the extraction rules are embedded in the underlying source codes. In other instances, a completely new language is created, such as WebL[5] by Compaq and WebQL[6] by QL2 Solution.

Firat's Matrix can best be summarized in the table below where he also gave examples of how the various academic tools fit under this matrix.

	Declarative		**Non-Declarative**	
	WAD	***WAS***	***WAD***	***WAS***
Manual		Mobie(Tsimmis)	Jedi	Araneus, WebL
Semi-automatic	NoDoSe	Caméléon	W4F	
Automatic	Lixto	WIEN, Stalker	XWrap	

Figure 3: Firat's Matrix

Source: Firat's PhD thesis

2.5 Laender's Taxonomy

Laender and Riveriro-Neto proposed a taxonomy to classify web wrapping toolkits based on toolkits' respective extraction techniques.[2] They divided web wrapping toolkits into six categories: language for wrapper development, HTML-Aware tools, NLP-based tools, Wrapper Induction tools, Modeling-based tools and Ontology-based tools. The following section briefly described the analysis that Laender & co. presented in their paper.

2.5.1 Languages for Wrapper Development

Apart from main stream programming language such as Perl and Java, specially designed programming languages have been developed to assist programmers in creating web wrappers more efficiently. As a programming language of its own, this type of web wrapping languages is usually non-declarative, i.e. their extraction extractions are hidden or embedded into the languages themselves. Examples of web wrapping languages include Jedi[7], TSIMMIS[8] and WebQL[6].

2.5.2 HTML-Aware Toolkit

HTML-aware toolkits treat web pages as a document tree, and rely on the underlying HTML hierarchical structure of the web pages for data extraction. This type of toolkits typically requires

additional component module to mitigate problems that arise from parsing web pages that do not conform to the HTML standard. Examples of HTML aware toolkits include XWRAP Elite[9] and Lixto[10].

2.5.3 NLP-based Toolkits

Natural language processing (NLP) is a technique for extracting data from free text documents where there is no structured presentation or layout. NLP-based toolkits make use of techniques such as filtering and tagging to establish relationship between phrases and sentences elements. Extraction rules are then derived from studying these relationships. Examples of NLP-based toolkits include RAPIER[11] and WHISK[12].

2.5.4 Wrapper Induction Toolkits

Typically, wrappers are automatically generated by feeding training examples to wrapper induction toolkits, for which delimiter-based extraction rules are derived. Toolkits in this category make use of the formatting features and hence the structures found on the documents for data extraction. Examples of this group of toolkits include WIEN[13] and STALKER[14].

2.5.5 Modeling-based Toolkits

Toolkits in this category create wrappers based on a pre-defined domain model, which specify a certain desired structure. The wrappers then extract data from web pages which exhibit structure similar to the domain model, in part or in whole. Examples of this type of toolkits include NoDoSe[15] and Robosuite[16].

2.5.6 Ontology-based Toolkits

Ontology-based web wrapping technologies are relatively new compared to most other web wrapping toolkits. Toolkits of this category make use of application ontology to locate data constants in the document. These constants are then used to construct objects. Using these objects, the application ontology can then be used to generate wrappers automatically. The pioneer effort of this group of toolkits comes from the Data Extraction Group[17] in Brigham Young University.

Chapter 3: Taxonomy of Web Wrapping Toolkits

Most of the taxonomies for web wrapping toolkits discussed so far employ a "top-down" approach in categorizing these systems. Using a set of pre-defined functional groupings, each toolkit is pigeon-holed into one or more of these groupings, sorted according to their respective characteristics. This approach has several merits. It is useful for providing an overarching framework in understanding the functional aspects of the web wrapping toolkits under study. It is also useful as a guide in determining the level of expertise required in the generation of wrappers. Finally, it provides a preliminary assessment of the functional effectiveness of each toolkit, through the study of such features as web document structure and the degree of "declarative-ness" of the extraction rules of the toolkits.

In the following section, a different way of categorizing web wrapping toolkits is introduced. Using a "bottom-up" approach, a two-tier approach into the categorization of web wrapping technologies was developed. A table was first constructed which listed all the non-trivial features of each and every toolkit, grouping these features into four layers: general, retrieval, extraction and conversion. Then, each of these four layers was further "streamlined" to obtain a set of core features that were considered to best describe the functionality of these toolkits. This "streamlined" table formed the 2nd-tier taxonomy.

3.1 The 1st-tier "Bottom-Up" Taxonomy

The 1st-tier taxonomy is shown in Table 1 on the following page. It lists most of the important features of each web wrapping toolkit that were covered so far. As can be seen from the table, it comprises of four layers: general, retrieval, extraction and conversion. The general layer lists some of the more general information about the toolkit, such as the degree of automation in terms of wrapper generation and the types of platform the toolkit supports. The retrieval layer shows the capabilities of each toolkit in accessing and retrieving web content. For example, it measures whether a particular toolkit has the ability to handle cookies and gaining entry to protected web sites. The extraction layer mainly displays the characteristics associated with the extraction rules of the wrappers, such as whether the wrappers support SQL, web crawling, scheduled extraction and so on. Finally, the conversion layer measures the mapping/connectivity features of the toolkits.

	Company	Fetch	MIT Sloan	ItemField	IBM	WebMethods	Shue Tech	IIQL Tech	Kapow Tech	ExtraData Tech	Altercept	Lencome Software	Lixto	Cannotate	Loton Tech	Thunderstone	QL2 Software	UC Dublin	TaskWare	Georgia Tech	XSB
	Tool	AgentBuilder	Cameleon	ContentMaster	DB2 IIC	Integration Platform	Mine the Web	IIQL	RoboSuite	Search-Extract	The Easy Bee	Visual Web Task	Visual Wrapper	vTag	WebDataKit	Webinator	WebQL	WIEN	WinTask	XWrap Elite	XRover
	General																				
1	Degree of Automation	Semi	Manual	Semi	Semi	Semi	Manual	Semi	Semi	Semi	Semi	Semi	Semi	Semi	Manual	Manual	Manual	Auto	Semi	Semi	Semi
2	HTML-Awareness	yes	no	yes	no	(no)	no	no	no	(yes)	no	(yes)	yes	no	yes	yes	no	yes	(no)	yes	yes
3	Platform	Windows	Windows	Windows, Unix	Windows, AIX, Unix	(Windows, Unix)	Windows, Mac OS	Windows	Windows, Linux	Windows	Windows	Windows	Windows	web-based	Windows	Windows, Solaris, Linux	Windows, Solaris, Linux	Windows	Windows	web-based	Windows, Unix
4	GUI	yes	(yes)	yes	yes	yes	yes	yes	yes	yes	yes	yes	yes	yes	yes	yes	yes	yes	yes	yes	yes
5	Editor	no	yes	yes	no	(no)	yes	yes	yes	yes	no	no	yes	no	yes	yes	yes	no	yes	no	yes
	Retrieval Features																				
6	HTML Headers	yes	yes	yes	yes	yes	no	yes	yes	yes	yes	no	yes	yes	(no)	yes	yes	yes	yes	yes	yes
7	HTTP Methods	yes	yes	yes	yes	yes	yes	yes	yes	yes	yes	yes	yes	yes	yes	yes	yes	yes	yes	yes	yes
8	Cookies Handling	yes	yes	yes	(yes)	(yes)	no	yes	yes	yes	yes	yes	yes	yes	(no)	yes	yes	(no)	yes	no	yes
9	SSL Support	yes	yes	yes	(yes)	(yes)	(yes)	yes	yes	yes	yes	yes	yes	yes	(no)	yes	yes	(yes)	yes	no	yes
10	Script Interpretation	yes	yes	yes	(yes)	(yes)	no	yes	yes	(yes)	no	yes	yes	yes	(no)	yes	(yes)	(no)	no	no	yes
11	FTP Support	(yes)	yes	(no)	yes	yes	no	yes	(yes)	no	no	no	yes	(yes)	(no)	(yes)	yes	no	yes	no	(no)
12	Input Format	text & binary	text & binary	text & binary	text & binary	text & binary	text, HTML	text & binary	text & binary	text, HTML	text, HTML	text, HTML	text, HTML	text & binary	text, HTML	text & binary	text & binary	text, HTML	text & binary	HTML	text, HTML
	Extraction Features																				
13	SQL Support	yes	yes	(yes)	(yes)	yes	no	yes	yes	no	no	yes	yes	yes	yes	yes	yes	(no)	no	yes	yes
14	Web Crawling	yes	yes	yes	yes	yes	yes	yes	yes	yes	yes	yes	yes	yes	yes	yes	yes	no	yes	yes	yes
15	Scheduled Extraction	yes	yes	(yes)	(yes)	yes	yes	yes	yes	no	yes	no	yes	yes	(no)	yes	yes	(no)	yes	(no)	yes
16	On-deamand Extraction	yes	yes	(yes)	yes	yes	yes	yes	yes	yes	yes	yes	yes	yes	yes	yes	yes	yes	yes	yes	yes
17	NLP Support	no	no	no	(yes)	no	no	yes	no	no	no	no	no	no	no	yes	(no)	no	no	no	no
18	Scripting Language		regular expression	Parser Script Language			Spider Scripting Language	NQL	regular expression	Perl, VBScript, Jscript			Elog		SQL for HTML and XML	Vortex, Regular Expression	WebQL		WinTask programming language	JAVA	Prolog
	Conversion Features																				
19	Output Format	XML	XML	XML	text	XML	text	XML	XML	text	text, HTML	text	XML	XML	text	text	XML	text	text, HTML	XML	XML
20	API	yes	JAVA, .Net	JAVA, .Net, COM	yes	yes	yes	yes	JAVA, .Net	(yes)	yes	yes	yes	yes	JAVA	yes	COM, JAVA,.NET, C++	no	yes	yes	yes
21	Database Driver	ODBC	(ODBC)	(ODBC)	ODBC, JDBC	ODBC, JDBC	no	ODBC, JDBC	ODBC	no	OLEDB/ODBC	(ODBC)	ODBC, JDBC	(ODBC)	JDBC	(ODBC)	ODBC	no	ODBC	no	ODBC

Table 1: 1st-tier Taxonomy

While the taxonomy was designed to be as comprehensive and exhaustive as possible, some information was not available, either because of proprietary information or the lack of product literature. In cases where explicit information cannot be obtained, derivations were made based on the behavior of the toolkits that were tested. Sometimes, when a particular product does not support a feature explicitly, but it provides a way of achieving the same task, it was considered to be supporting that particular feature. For example, while WebQL studio does not support script interpretation directly, a developer of WebQL studio has the option to code a WebQL wrapper that is capable of imitating the functions of scripts. In cases like these, a bracket is used to indicate that particular attribute as "conditional". The next section provides a brief description of its features and its corresponding significance to the taxonomy.

3.1.1 Degree of automation

The degree of automation refers to the ways in which wrappers are generated by their respective toolkits. As discussed in chapter 2 previously, it could mean whether a programmer is needed to manually write the codes, or whether the toolkit is able to generate wrappers semi-automatically through demonstrations, or whether training examples need to be supplied to the toolkit for wrappers to be generated automatically using machine-learning algorithms or other artificial intelligence methods.

3.1.2 HTML-Awareness

HTML-awareness is a term used to describe how effective a particular toolkit is in extracting data from HTML-based web pages. In general, toolkits that make use of the HTML hierarchical tags in web pages to parsing are said to be more HTML-aware. However, it is not equivalent to saying that toolkits with a higher degree of HTML awareness are superior to the others. In fact, due to the relatively small number of web pages that make use of this HTML hierarchical tags (estimated to be 20% of total web pages on the Internet), the effectiveness of these type of toolkits are severely curtailed by the need to introduce error-correction mechanism when wrapping non-HTML compliant web pages.

3.1.3 Platform

Platform refers to which operating system the toolkit supports. For the purpose of this thesis, no effort is made to distinguish between the variants within a particular operating system, i.e. Win NT and Win XP are both considered as "Windows".

3.1.4 GUI

GUI stands for Graphical User Interface, and is almost ubiquitous for all current-generation toolkits. In the early days of web data extraction, however, most wrappers were generated through hand-written codes in the command prompt. There are still some legacy systems today that make use of command lines rather than GUI, although they are fast approaching extinction.

3.1.5 Editor

Editor provides a mean for the user to code a wrapper manually, and is a feature present in most toolkits that require manual generation of wrappers. Most semi-automatic toolkits, however, also include this feature into their program. This is to provide users with an additional avenue to wrap a complex web site, should the extraction of web data through demonstration prove to be too difficult to accomplish.

3.1.6 HTML headers

A good web wrapping toolkit must be capable of supporting a range of HTML headers. These HTML headers contain meta-information about the web pages and often also embody such features as automatic refreshes and redirection, common features found in a web browser. Some examples of HTML header tags include: <base>, <link>, <meta>, <title>, <style>, and <script>. The ability to support HTML headers is important to all types of toolkits as this feature essentially specifies how web pages can be retrieved. The lack of this feature limits the accessibility of wrappers to certain web pages.

3.1.7 HTTP methods

Like HTML headers, HTTP methods such as GET and POST deal with the retrieval process of a wrapper. HTTP methods are needed in order for one application, like a web browser or a wrapper, to talk to another application via the world-wide-web. All web wrapping toolkits need to have this feature in order to access any web page. Naturally, all the toolkits shown in Table 1 support this feature.

3.1.8 Cookies handling

A cookie is a piece of text that a web server uses to store on a user's hard disk. It allows a web site to store information on a user's machine and to retrieve it later. The ability to handle cookies effectively will affect the performance of a web wrapper directly. Web wrappers that are not anointed with cookies handling capability will not be able to access some web sites where cookies recognition is required. Future wrappers developers should take this into consideration as the number of web sites built around simple HTML structure begins to dwindle and demand for cookie-enabled web sites rises.

3.1.9 SSL support

The Secure Socket Layer (SSL) is a technology used to protect web communications. The SSL security protocol provides data encryption, server authentication, message integrity and other web security protection measures. Toolkits that support SSL are capable of extracting data from SSL-protected web sites, such as from online bank accounts.

3.1.10 Script Interpretation

The exact definition of script is often fuzzy, but most people tend to agree that it is a small program that executes itself on a client's machine when its associated web document is loaded, or whenever it is manually activated through clicking. Most scripts are written in either JavaScript, VBScript or PHP. Scripts provide an interactive experience for the end user of a web document, but most wrappers nowadays lack the ability to decipher or interpret scripts well enough. Therefore, from the point of view of a toolkit, the ability to interpret script leads to a greater number of accessible web sites.

3.1.11 FTP support

Toolkits that offer FTP support are able to navigate not only http sites, but also ftp sites. FTP, of course, stands for File Transfer Protocol and is the de facto standard used for exchanging and storing large documents online.

3.1.12 Input format

Input format refers to the type of data format that the wrappers can extract. Most wrappers are capable of extracting plain text and HTML web pages, while the more advanced wrappers are able to extract binary files such as JPEG files. As expected, the more input formats that a wrapper can support, the more value the toolkit has to offer as the input options expand.

3.1.13 SQL Support

SQL stands for Structured Query Language and is an industry-standard language for creating, updating and querying relational database systems. A toolkit that supports SQL also allows the users to send queries to the web sites as if they are relational databases. For example, users can perform a "joins" operation between multiple web sites if a SQL-enabled web wrapping toolkit is used.

3.1.14 Web Crawling

Web crawling refers to the ability of the wrappers to navigate their ways from one web page (known as the "source" page) to the other (known as the "target" page), following links and filling up forms in the process. Web crawling is an important characteristic of web wrappers. Less robust wrappers are not able to follow links and therefore will never reach the "deep web", or web pages that are more than a few clicks away from an originating "source" page. On the contrary, advanced web wrappers are even able to fill up forms and are widely used to wrap pages that require a user to enter username and password, such as a web-based email client like Yahoo Mail.

3.1.15 Scheduled extraction

Scheduled extraction describes the ability of the wrappers to automatically extract data from the targeted information source on a periodical basis. This feature is especially useful for wrapping web content that changes very frequently, such as a news site.

3.1.16 On-demand extraction

On-demand extraction works in tandem with scheduled extraction. While scheduled extraction frees the user from needing to visit the sites regularly for information extraction, on-demand extraction gives the user the ability to visit a web site and mine for information whenever he wants. Most wrappers, if not all, have this feature enabled.

3.1.17 NLP Support

Natural Language Processing or NLP-based toolkits are especially suitable for extracting data from documents where there are a lot of free text content such as advertisement and news reports. They make use of techniques such as filtering and part-of-speech tagging to derive extraction rules from phrases and sentences. Because most of the web documents exist in semi-structured formats, NLP-based toolkits are not widely accepted by the web wrapper community. Representative examples include RAPIER and WHISK.

3.1.18 Scripting language

Unlike section 3.1.10 where a script is interpreted as a small executable program that runs on a client's machine and script interpretation describes the ability of web wrappers to understand these programs, scripting language refers to the method used by the wrappers in extracting information from the web. It might involve pattern matching methods such as regular expressions, or a specialized extraction language such as WebL or ELOG, or a combination of both. Scripting languages are usually written in the editor to specify the extraction rules. In general, they are more prevalent in toolkits where wrappers have to be manually generated.

3.1.19 Output Format

Output format refers to the type of output that is transformed by the wrappers. For simplicity, most toolkits set the default output type to XML, to take advantage of its flexibility to map into other data formats. Toolkits that offer more than XML make the conversion from HTML text to other desired formats easier and quicker, whereas toolkits that support singular data output format such as plain text require additional processing if more sophisticated output format is required.

3.1.20 API

API stands for application program interface and is defined as a formalized set of software calls and routines that can be referenced by an application program in order to access supporting network services. [6] It provides an interface for an application to be linked to the other, usually

via one or more high-level programming languages such as JAVA or C++. Toolkits that support a wide range of APIs will benefit from other "add-on" features provided by third-party vendors. The real value of API on a toolkit therefore lies with its extended value-adds.

3.1.21 Database Driver

The ability of the toolkit to integrate and work well with other applications is crucial in the mapping stage. Toolkits equipped with a wealth of database drivers offer a wider range of options to connect to other database-compliant software packages, such as Excel. Due to the popularity of ODBC, toolkits that only support ODBC drivers are often considered to be highly "connected" since most other software applications are ODBC compliant, if they support database connectivity. ODBC stands for Open Database Connectivity and is a widely accepted application programming interface (API) for database access.

3.2 The 2nd-tier Taxonomy

As mentioned previously, the 2nd-tier taxonomy is a "streamlined" version of the first, and a continuation of our "bottom-up" approach to classifying web wrapping toolkits. In this taxonomy, a set of core features is selected from each of the four layers in the 1st-tier taxonomy shown in Table 1. The features in these layers represent the core functionalities that full-scale web wrapping toolkits usually support. For instance, script interpretation is included in this taxonomy because a toolkit capable of interpreting script languages are important to the overall user experience in terms of accessing and retrieving web content. Similarly, a wrapper that is able to automatically extract information from the web on a periodical basis is important to the web extraction experience, and represents an approach consistent with the future direction of the web wrapping technologies. On the other hand, NLP support is excluded in this taxonomy because it is a feature not widely adopted by most web wrapping toolkits. In Table 2 below, the "streamlined" 2nd-tier taxonomy is presented. For ease of readership, a number at the bottom of each column in the table is provided to show the number of features supported by the corresponding toolkit (out of a total possible of 12). Note that for comparison purposes, two out of the fourteen parameters are excluded from the final tally (and hence the number 12). They are: the degree of automation and HTML-Awareness, which are actually properties rather than features of the toolkits.

		Agent Builder	Cameleon	ContentMaster	DB2 IIC	Integration Platform	Mine the Web	NQL	RoboSuite	SearchExtract	The Easy Bee	Visual Web Task	Visual Wrapper	vTag	WebDataKit	Webinator	WebQL	WEN	WinTask	XWrap Elite	XRover
1	Degree of Automation	S	M	S	S	S	M	S	S	S	S	S	S	S	M	M	M	A	S	S	S
2	HTML Awareness																				
3	HTTP Methods																				
4	HTML Headers																				
5	SSL Support																				
6	Cookies Handling																				
7	Script Interpretation																				
8	SQL Support																				
9	Web Crawling																				
10	Scheduled Extraction																				
11	On-demand Extraction																				
12	Ouput Format (XML)																				
13	API																				
14	Database Driver																				
		12	12	12	11	12	6	12	12	8	9	9	12	12	6	11	12	4	9	7	12

Table 2: 2nd-tier Taxonomy

From the table above, 10 out of the 20 toolkits surveyed support a comprehensive range of features from HTTP methods, cookies handling to database connectivity. For simpler illustration, some of the granularity indicated in the 1st-tier taxonomy is omitted. Therefore, instead of specifying the types of database driver a particular toolkit supports, a toolkit is considered as having database connectivity feature whether it supports ODBC, JDBC, or both.

Chapter 4: Analysis of Web Wrapping Toolkits

Using the set of parameters from the taxonomy, a comparative study was performed to analyze the 20 toolkits that were studied. In this chapter, an explanation to the software architecture behind each of the toolkit that will be presented, including how the toolkit operates where appropriate. A description of the toolkits with respect to the taxonomy will also be given, including comparison with other toolkits that are included in the taxonomy. In the following section, toolkits that have been tested are underlined.

4.1 AgentBuilder[18]

AgentBuilder is a semi-automatic web-wrapping toolkit that supports a comprehensive range of features and functionalities including cookies, SSL, deep web links, HTML text and other data types such as image files. Like most other toolkits, AgentBuilder attempts to differentiate itself from competition by focusing on its proprietary extraction algorithm. According to Fetch Technologies, the maker of AgentBuilder, the extraction rules that underpinned AgentBuilder are based on "landmarks" or "a group of consecutive tokens". These landmarks are then used by the web wrappers to locate the start and end of fields within a given page. Using a hierarchical induction algorithm, AgentBuilder is able to parse a given web page into a document tree, breaking down complex web pages into small segments and hence allowing wrappers to perform the extraction tasks more efficiently.

The most promising aspect of this toolkit, however, lies in its mapping/mediation layer. Using a machine-learning approach, AgentBuilder is able to address the many problems associated with record linkage. Record linkage is a fundamentally issue that arises in the integration of multiple data sources, across different platforms. For example, connecting two databases causes a record linkage problem if one of the databases uses the American way of recording date (month/day/year) whereas the other employs the European convention of day/month/year. By automating the recognition of these differences using machine-learning algorithms, AgentBuilder is able to reconcile and correct these differences automatically without the need for further manual programming. This is of particular relevance to web wrapping solutions where data from disparate sources often have to be combined and integrated in order to produce a coherent output.

All in all, AgentBuilder has a comprehensive range of extraction features, with a rather unique mediation feature through its machine-learning approach to record linkage problems. By converting all output data into XML, AgentBuilder also offers an added flexibility for users to customize their eventual display options.

4.2 Cameleon[3]

The latest version of Cameleon was developed in C#, taking advantage of the extensive .NET library. As such, it renders Cameleon a Windows-based software, although its predecessor, which was written in JAVA, was capable of running in multiple platforms.

Cameleon supports almost all of the features that a comprehensive web client utilizes. For example, it is able to handle such functions as SSL, cookies, authentication, HTML headers, HTTP methods and script interpretation (including javascript). It takes in input in both text and binary formats, and therefore is capable of covering a large range of data types. As a WAS-based web wrapping toolkit, Cameleon does not rely on the underlying HTML hierarchical tags for data extraction. Instead, it makes use of regular expressions in formulating the extraction rules, which ultimately determines the behavior of its wrappers. The SQL support it provides renders a web effectively a relational database, and hence allowing users to directly query the web using normal SQL expressions. Apart from not supporting natural language processing, Cameleon is otherwise robust enough to handle web crawling and FTP access, in addition to scheduled and on-demand extraction of its wrappers (although scheduled extraction is not part of the core Cameleon package). It provides a XML output format, a range of APIs that include JAVA and .NET and an ODBC driver that allows for extended database connectivity with other software applications.

In terms of the architecture, Cameleon is made up of three core modules that include a query handler, an extraction and a retrieval module, as illustrated in Figure 4.1 below. To access Cameleon, a user typically sends SQL queries through the graphical relational front end of Cameleon, or more commonly, through HTTP. Together with the Planner Optimizer Executioner (POE), the relational front end allows common SQL operations such as “joins” to be performed between different web pages and databases. The queries sent by the users are then handled by the query handler that first looks into the registry to determine which wrapper needs to be retrieved and which attributes in the wrapper are relevant. The wrapper is then passed on to the extraction module in which extraction rules to data streams are applied. Finally, the modified wrapper passes through the retrieval module where real web data extraction takes place. Here, Cameleon makes use of the embedded web client in the .NET library that supports a wide range of features such as authentication and cookies handling as mentioned before.

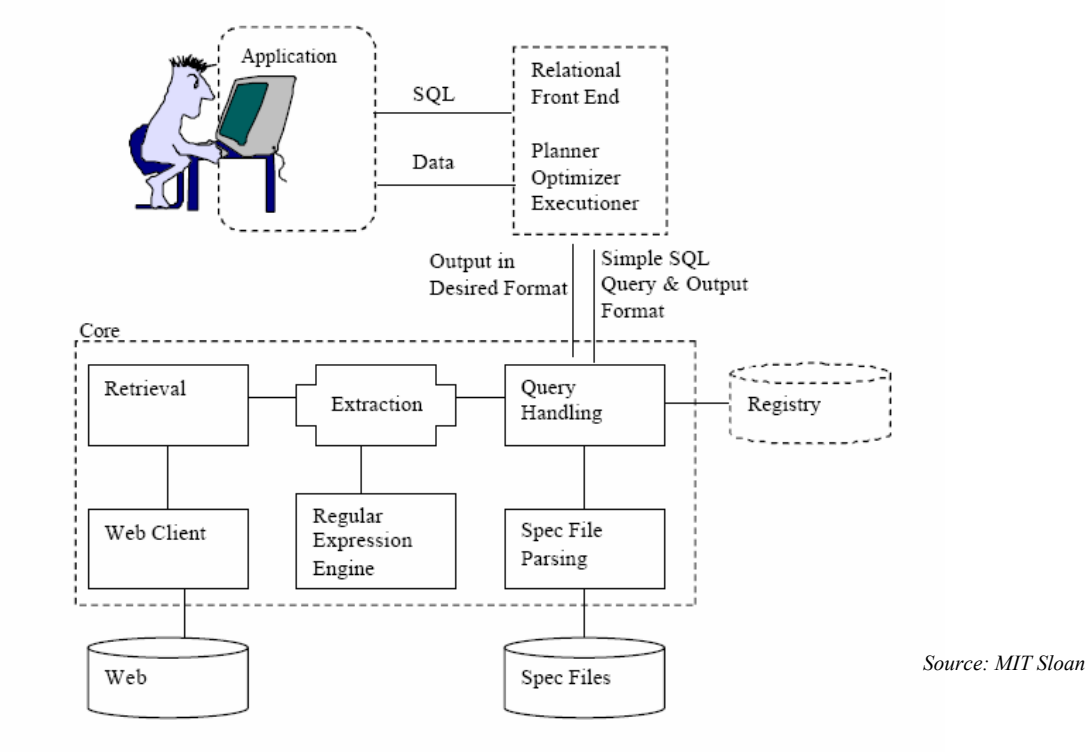
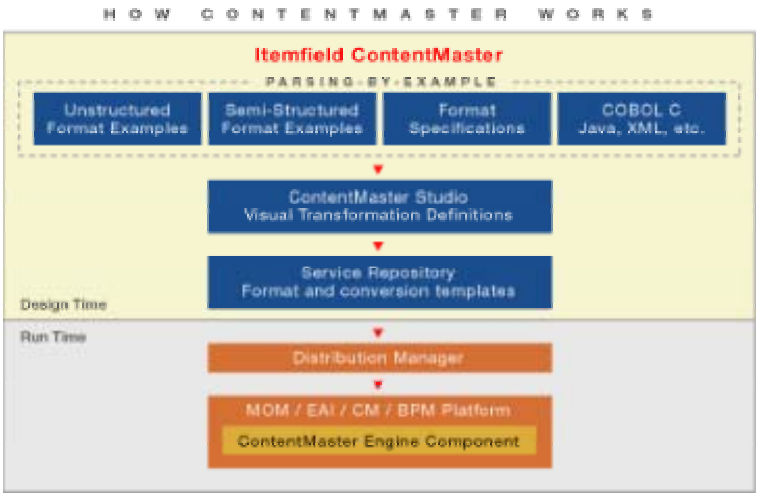


Figure 4: The Architecture of Cameleon

4.3 ContentMaster[19]

ContentMaster continues the trend in industry towards automating generation of web wrappers by providing a point-and-click visual interface through its ContentMaster studio module. This visual environment not only guides a user in creating web wrappers, but also provides the option for the user to define bi-directional data transformation between different kinds of data formats. For example, ContentMaster can be used to build a parser that transforms data from a certain data source into XML, and a serializer that does the exact opposite, i.e. converting XML into a source data. It is worth noting, however, that the source data converted from XML needs not be the same as the original data format from which that XML is converted.

Essentially, ContentMaster comprises of two separate modules: ContentMaster studio and ContentMaster engine. As discussed above, the studio module automates the creation of scripts to allow for bi-directional data transformation. These scripts are then moved to the service repository where the format and conversion templates (i.e. the wrappers) are stored. At run time, these scripts or wrappers are executed by the ContentMaster engine which also allows for the scripts to be embedded in major EAI or MOM platforms. Figure 5 shows the architecture of ContentMaster, including a parsing engine at the top that supplies text-based documents to ContentMaster studio.



Source: Itemfield
www.itemfield.com

Figure 5: ContentMaster's architecture

Apart from its data transformation capability, ContentMaster also offers a powerful retrieval layer. It meets all the requirements under that category in Table 1, which means the wrappers it generates can be used to automatically log on to web sites, process scripting languages, access secured web pages, handle cookies and extract data from dynamically generated web pages. It is however unclear if ContentMaster offers the capability to allow a user to query the web directly via a SQL interface. Nevertheless, the inclusion of an editor allows a user to define and write a wrapper from scratch and thus providing possible opportunities to offer similar query capabilities. As a strong propellant of data transformation and connectivity, it is no surprise to find that ContentMaster also offers one of the more complete range of APIs including JAVA, .NET and COM interfaces.

4.4 DB2 Information Integrator for Content[20]

DB2 Information Integrator for Content, abbreviated to IIC, is an integration platform developed by IBM. Unlike most of the toolkits we have discussed so far, IIC is primarily an integration platform that connects disparate data sources within a large enterprise. For example, it is ideal for connecting a variety of content severs or data sources such as, say, an Access Database and an IBM DB2 system.

DB2 Information Integrator for Content is made up primarily of about 5 core modules that allow a user to access this technology through a portal or a simple web browser such as Internet Explorer or Netscape Navigator. The overall architecture of IIC is depicted in the following diagram below:

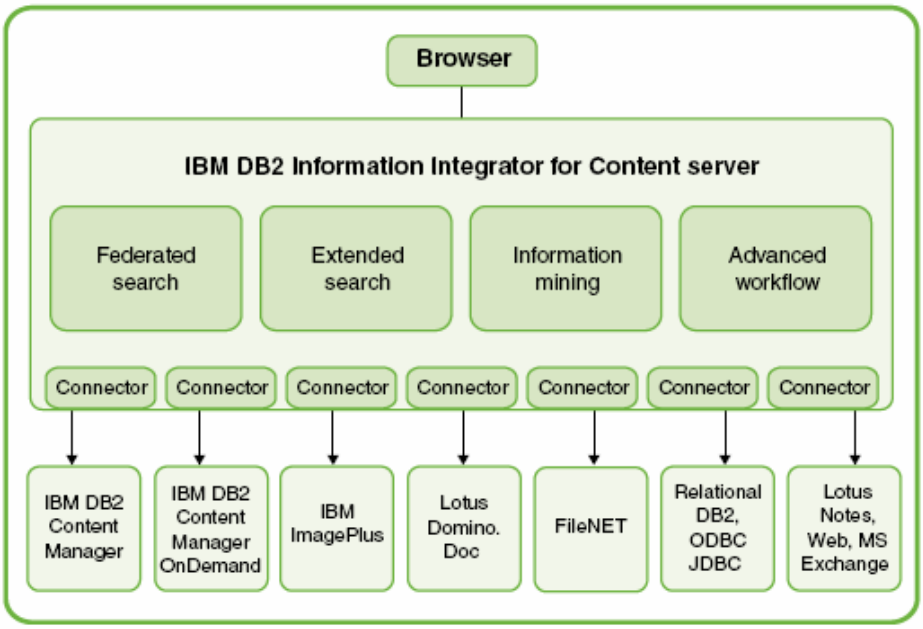


Figure 6: Architecture for DB2 Information Integrator for Content

Source: IBM www.ibm.com

As shown in Figure 6 above, IIC is accessible through a portal or a web browser, which in turn is linked to the following modules:

1. The federated server – this module allows users to access diverse content and data sources, and it includes intelligent text and image queries across multiple repositories.
2. Extended Search server – this sever is responsible for conducting federated queries that run on both internal file system and the Internet.
3. Information mining toolkit – this module provides an integrated and organized access to documents scattered across various content servers or data sources.
4. Advanced workflow – the primary function of this module is to manage the content lifecycle.
5. Connectors – the various connectors are used to link the backbone architecture to other data sources of interest so that the backbone modules mentioned above can access the underlying data sources transparently.

The IIC is one of the most expensive solutions in the market, and it is therefore not surprising to find that it contains most of the features commonly found in a web wrapping toolkit. For example, it supports many of the major features such as SSL, cookies, script interpretation, SQL support and so on. However, it remains as an integration platform to connect disparate data sources from legacy systems, and as such, places less emphasis on web data extraction (from Figure 1, only one connector is linked to the Internet). This is evident from its lack of an editor that would allow users to write customized wrappers to extract content from the Internet. Nevertheless, IIC is outstanding in terms of its database connectivity features. It has wide support for many database applications including DB2, Oracle, Sybase, MS SQL Server, Informix and many other ODBC and JDBC compliant packages.

4.5 Integration Platform[21]

WebMethods Integration Platform provides an infrastructure onto which organizations can use to connect heterogeneous data sources that reside on disparate systems. It is able to integrate J2EE, .NET and legacy systems, thereby making possible the linking of business processes, enterprise systems, databases, workflows and web services.

Integration Platform is equipped with an impressive portfolio of integration modules that together support web services, J2EE and .NET integration, legacy systems and adapters. This provides a scalable XML-based environment that enables the integration of enterprise applications between two parties through the Internet. One classics example in which such integration platform will come in useful is in the area of e-procurement or electronic data interchange (EDI) in which 2 or more parties have to communicate with one another through a common channel. Integration Platform provides such a channel to facilitate the transfer of information between these parties through its XML-based environment. The architecture of such an integration platform among trading partners can be illustrated in Figure 7 below:

Source: WebMethods
www.webmethods.com

Figure 7: Rapid Integration based on open standards

As an integration technology-focused solution, it is perhaps no surprise that WebMethods Integration Platform also comes with ODBC and JDBC database drivers, which, together allow this toolkit to link up with database systems such as Oracle, SQL server, Informix, Sybase and DB2. Its API is also comprehensive enough to handle applications written in C/C++, COM, JAVA/J2EE, EJB and of course, .NET. Through its XML-based output format, WebMehthods Integration Platform is able to offer additional flexibility in terms of the eventual output formats. In terms of the extraction features, this toolkit provides enough functionality for effective extraction. This includes the support for SQL, web crawling, FTP and scheduled extraction. The combined benefits of this range of extraction is the ability on the part of the user to generate wrappers that can take in queries from the user and navigate deep into the web from a given URL, treating the Internet as if it is one giant relational database.

4.6 Mine the Web[22]

As the name suggests, Mine the Web or MTW parses the web to retrieve information, before storing the retrieved content into a pre-selected database defined by the user. Mine the Web is one of the few commercial web wrapping toolkits that relies purely on manual generation of wrappers, or spiders, that are then used for web data extraction. Running on both Windows and Mac OS platforms, MTW uses a proprietary scripting language known as Spider Scripting Language for wrappers generation. While this spider scripting language does not necessarily make use of the underlying HTML tag hierarchy for data extraction, it depends heavily on HTML tags for identifying the exact location of the data. Below is a small excerpt of a wrapper using the Spider Scripting Language:

```
1. startafter:
2. width=180 height=34></a></td></tr>|
3. endat:
4. <tr align=right><td align=left nowrap><a href="/q?d=t&s=^TNX">|
```

Notice that the program specifies where the extraction is supposed to start at line one and where it should stop at line 3. At lines 2 and 4, related HTML source codes of the web page are supplied. It is therefore implicit that the wrappers hence generated are not likely to work well on other type of data sources that do not make use of the HTML tags. For example, these wrappers are not able to extract data from say, a Word document or a PDF file.

Although MTW does not provide an ODBC or equivalent database driver, it is able to store the extracted data into tabular formats and into databases, making later retrieval quick and easy. However, its lack of support for SQL also means that unlike other competing toolkits, the user cannot interactively query the web as if it is a huge relational database. MTW also comes out light in terms of its retrieval functional layer. Despite of its manual nature of generating web wrappers, MTW still does not support many of the more advanced features. For example, it is not able to access SSL-protected web sites nor is it able to handle cookies or interpret scripting languages embedded in a web site. Nevertheless, for most simple display of real-time information on a web page, MTW is considered to be adequate.

4.7 NQL[23]

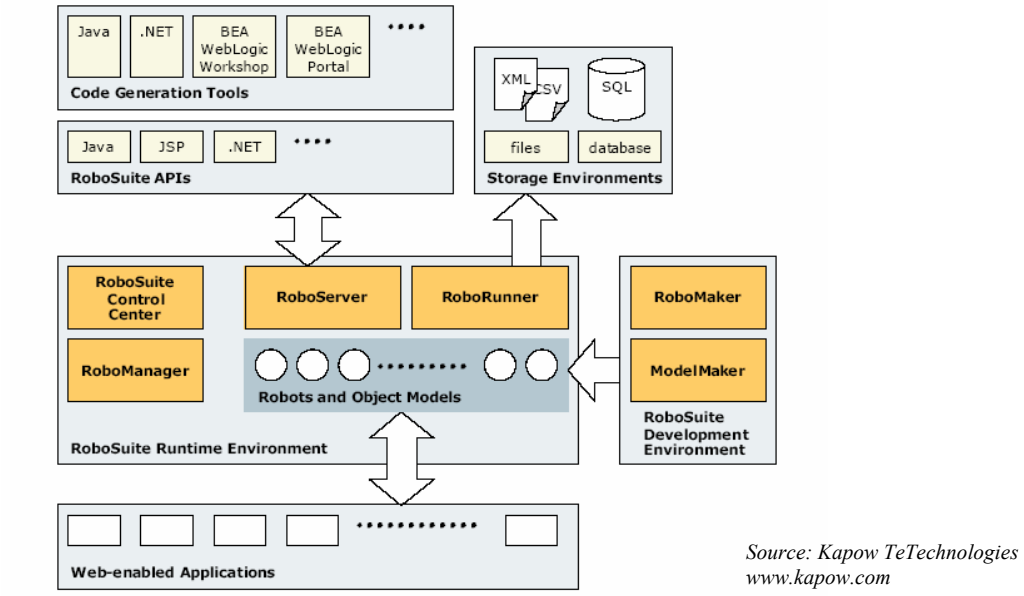
NQL stands for Network Query Language and, as the name suggests, is a SQL-like scripting language that is used to create connected applications such as web wrappers, or more commonly known as robots or intelligent agents. NQL the toolkit comes with a version that includes a browser recorder to guide a user in creating web wrappers using a visual environment. However, for the most part, NQL is considered a manual web wrapping toolkit where the wrappers are manually generated using the proprietary Network Query Language.

As with most manual web wrapping toolkits, NQL is expected to be powerful because the wrappers it generates often contain a high degree of customization that a developer or a user can harness through creative coding. And indeed, NQL does come with a generous host of features not commonly found in other toolkits. For example, NQL is equipped with artificial intelligence capabilities that includes neural networking, Bayesian inference, pattern matching and fuzzy logic – essential ingredients to creating robust web wrappers capable of extracting data from free text, semi-structured and structured data sources. With these features, NQL becomes one of the few commercial web wrapping toolkits that offers natural language processing capabilities. These artificial intelligence features, along with other building blocks of NQL such as the usual retrieval, extraction and mapping functions, can be entirely written using its own scripting language, also called Network Query Language, or NQL.

NQL the toolkit supports both ODBC and JDBC database drivers, allowing for a wider reach of database connectivity with other software applications. It also comes bundled with an API and is capable of converting extracted data of varying formats into XML, which can then be exported into other data formats easily. By offering a scheduled extraction function, web wrappers created using NQL can be made to perform "silent" extraction behind the scene, freeing the user from having to execute the wrappers constantly.

4.8 Robosuite[16]

Robosuite is one of the more complete solutions that were surveyed. It acts as a middleware between a client application and the source data from the web by providing a non-intrusive environment for a user to access, gather and mediate web data. Its architecture is based on a central server design in which an integration application, called RoboServer, is responsible for processing requests from client applications before executing the wrappers, called robots in this case. Through its extensive APIs, Robosute provides multiple interfaces for various third-party applications to interact with it. The simplified architecture diagram for Robosuite is provided below in Figure 8.



Source: Kapow TeTechnologies
www.kapow.com

Figure 8: How Robosuite works

4.8.1 Robosuite's Architecture

The robots in Robosuite are the web wrappers and they are primarily responsible for executing the commands from the RoboServer, which passes a set of objects to the robots as input. An object is a collection of attributes, and each attribute is associated with an attribute name and sometimes a single attribute value. The following example perhaps best captured what an object is:

"Each news object has attributes with attribute names such as headline, body text, date, author, etc., and each outputted news object will have different attribute values for each attribute (unless, of course, the same news object is outputted more than once!). An outputted object is called a returned object."[24]

There are many functions that an object can take on, but most of the times, they are used as data carriers, usually as input to a robot in the form of input objects, or alternatively as output from a robot in the form of output objects or database output objects. Objects are organized into domain models where each domain model is used exclusively to model a particular real-world event, such as news. The application that is used to create and hold these domain models in Robosuite is known as the ModelMaker. Figure 9 explains the relationship between objects, robots and the rest of the applications in Robosuite. According to this diagram, ModelMaker is used to create object model that comprises of a number of objects. These objects are then used by the RoboMaker as input to create robots. Although not shown in the diagram, objects can also be used as data carrier carrying the output from the robots.

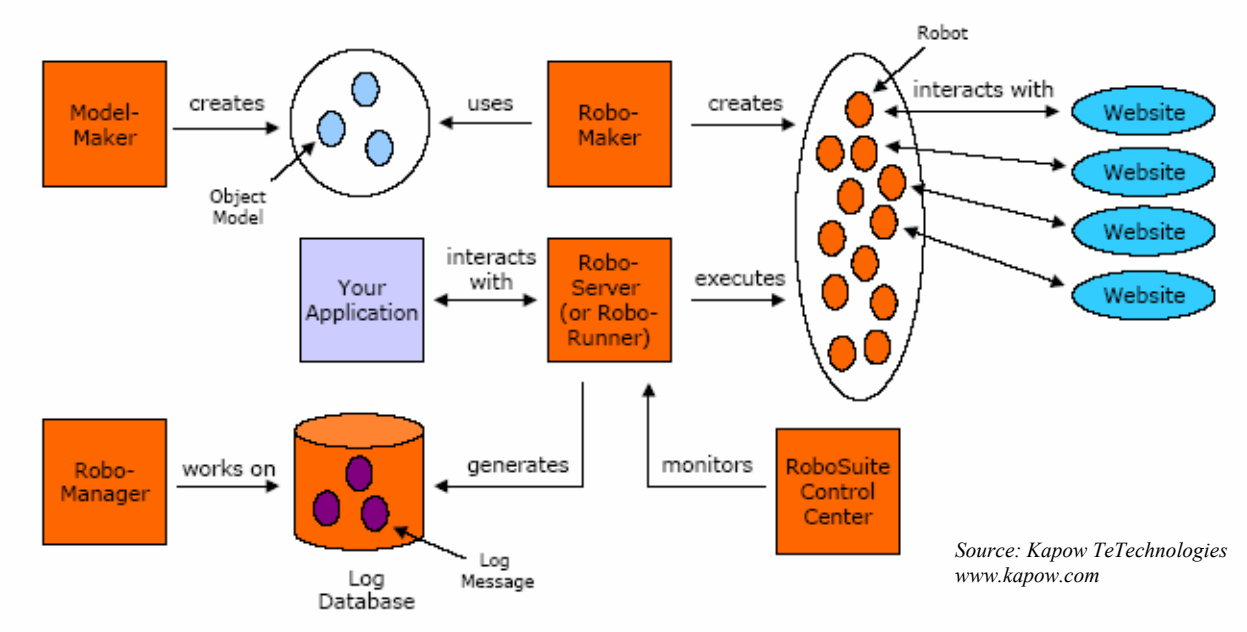


Figure 9: Relationship between the modules in RoboSuite.

Using the objects created in ModelMaker, robots are then generated inside RoboMaker. The following diagram depicts the relationship between the ModelMaker and the RoboMaker, plus additional elements of Robosuite. Figure 10 shows an example of a web-clipping robot. Note that the boxes at the top indicate the various steps taken by the robot whereas the clip browser window shows the source of the web data.

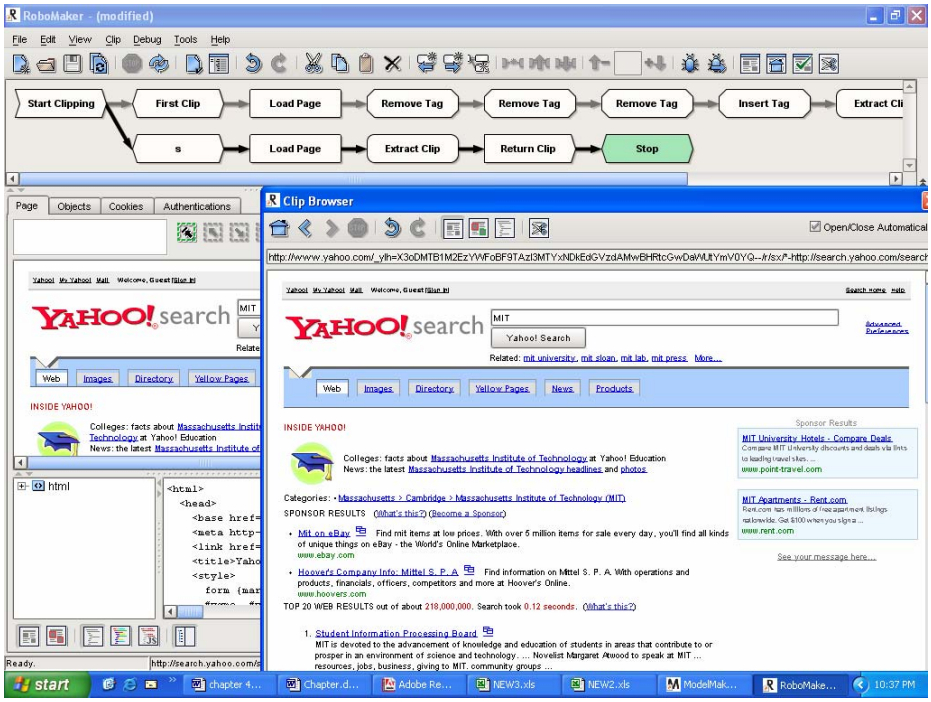


Figure 10: The RoboMaker

The two key features of Robosuite are its data collection and web clipping capabilities. In Robosuite, web clipping means gathering small segments (or "clips") of web pages from different sources into one single page. Using ModelMaker to create a domain model and then an object, a user can proceed to create one or more robots using RoboMaker. These robots are then sent out on a periodical basis or on an on-demand basis to collect information from those web sites predefined by the users. Because these robots do not make use of the underlying HTML hierarchy structure to parse web document into a tree, it is able to extract and collect data from more sources. The range of web sources for which these robots can extract data from expands further with its added capabilities to handle HTML headers, cookies and SSL, on top of its ability to interpret script languages such as javascripts.

Apart from a rather robust retrieval functional layer, Robosuite also provides a range of features to its extraction layer. For example, its support for SQL allows a user to query the web as if it is one giant database. Its web crawling feature allows the robots to venture further into deeper web in search of more accurate information. And just in case some web pages are to complex for the user to use the visual graphical environment for data extraction, Robosuite comes with an editor that allows him to manually code the robots using regular expressions. However, Robosuite falls short in its ability to handle natural language processing. Its ability to handle ftp sites is also limited to the ones with a html interface, and the robots are not yet robust enough to download files from ftp sides. Nevertheless, Robosuite again scores high in the mapping layer, possessing an ODBC database driver, JAVA and .Net APIs, and a XML output format. These features allow Robosuite a high degree of connectivity with many other third-party applications.

4.9 SearchExtract[25]

In comparison to most commercial web wrapping toolkits, SearchExtract is a streamlined cousin of its competitors. This is mainly because of its almost exclusive focus on tabular data extraction. SearchExtract does a pretty decent job in extracting web data that are hierarchical in nature, i.e. table, lists, etc. However, it is not able to extend this extraction function to other web data formats, making it rather limited in terms of its reach for web data. On top of that, it lacks the ability to parse through multiple pages, and does not support ODBC or other major database driver. This is in spite of its rather comprehensive retrieval layer which covers most of the major features such as support SSL and cookies handling.

SearchExtract does not offer support for SQL, which means that users are unable to send queries to the web via this toolkit. It also does not seem to have the ability to run the wrappers on a pre-defined schedule. SearchExtract does however offers some degree of customization in that it provides an editor in which user can manually generate a wrapper using some of the more common scripting languages. These scripting languages include Perl, VBScript and Jscript.

In short, Search Extract is a simple tool ideal for extracting static HTML data that are presented in an organized manner, such as a table or a list.

4.10 The Easy Bee[26]

The Easy Bee is a simple web wrapping toolkit in which its wrappers are entirely created through its visual interface. This toolkit has an architecture that makes use of a domain model, which is then used as the base for creating wrappers for data extraction in the world-wide-web. However, to avoid using too much technical jargons, Altercept, the maker of The Easy Bee, decided to use a naturalist metaphor to disguise the technicality. Thus, the domain model is known as the "Honeycomb" where the "Bees" or wrappers "live".

Using The Easy Bee proves to be "easy" indeed. To generate a wrapper (bee), a user simply needs to create a "hive" first where the future "bee" will live. To do this, he goes to the left-pane of the application known as the "Honeycomb", right-click on the sample honeycomb and selects "New Honeycomb" as shown in the Figure 11. A new Honeycomb will then appear and user can then add multiple "bees" or wrappers by selecting "New Honeybee" from the menu. After that, a user only needs to complete 3 easy steps in generating a fully functional bee. These steps are listed below:

1. Navigation – user enters the URL of the originating web page and navigates to the desired page.
2. Extraction – user selects a segment of the web page in which he wants to extract using a "point-and-click" method. Further refinement can be made via a selection parameter
3. Schedule – user decides when the bee will start to "fly" or extract. The current selection dictates that the bee can either fly at logon, at regular interval or on demand.

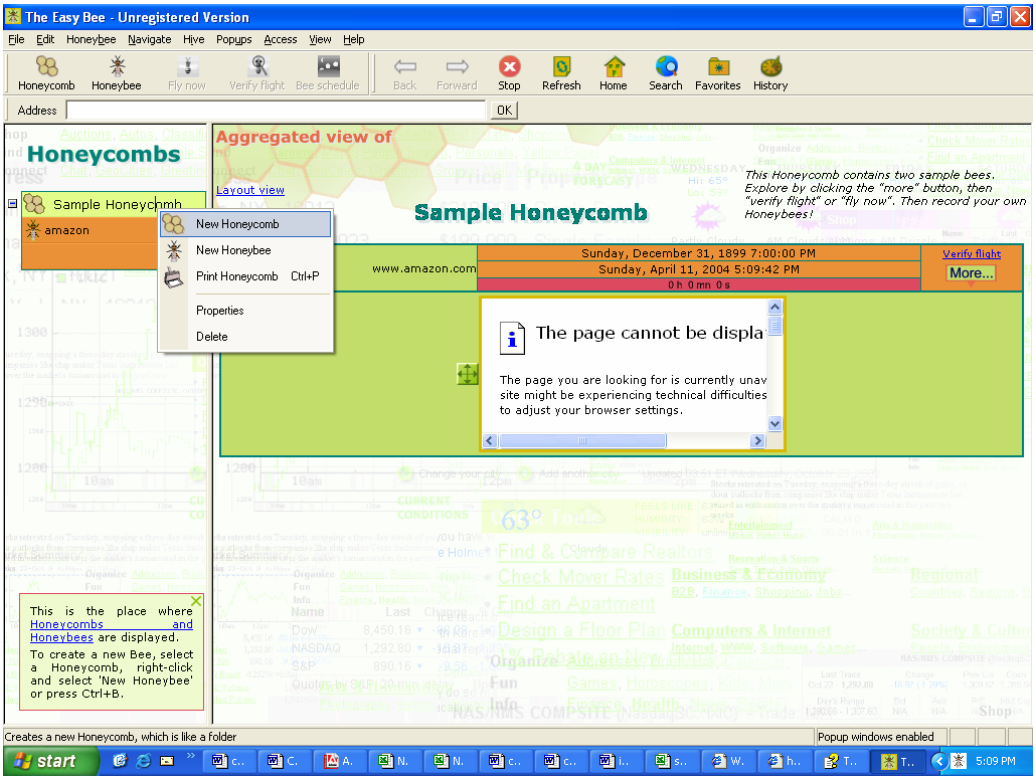


Figure 11: Creating a Honeycomb

The Easy Bee also provides a very practical function through its "private bee" and locks. It automatically recognizes and assigns a lock to a bee, making it a "private bee" whenever a login session is executed during the navigation step. For example, if a user wants to create a bee for his Yahoo Mail account, he would have to enter his username and password during the navigation step. The Easy Bee captures this login information, recognizes it, and automatically makes the bee created a "private bee". To access a private bee, user needs to supply a global password to unlock the bee. This global password is universal and can be used to unlock all the bees contained within the toolkit. In a way, this feature is most akin to online account aggregation service such as Yodlee, where a user only needs to remember one global password to assess his other password-protected accounts, if he has previously nominated these accounts. Figures 12 and 13 show how access control to the private bee is governed by a global password:

Despite the easy-to-use interface, The Easy Bee falls short on several fronts. First, it does not have as comprehensive a range of features as the other commercial web wrapping toolkits. For example, in terms of the retrieval layer, The Easy Bee does not support script interpretation. It has no SQL interface and does not allow a user to treat the web as a relational database by sending queries. While it offers a OLEDB/ODBC database driver, it has yet to be fully implemented, and the HTML-based output format that it supports limits its ability to easily transform the data for other uses.

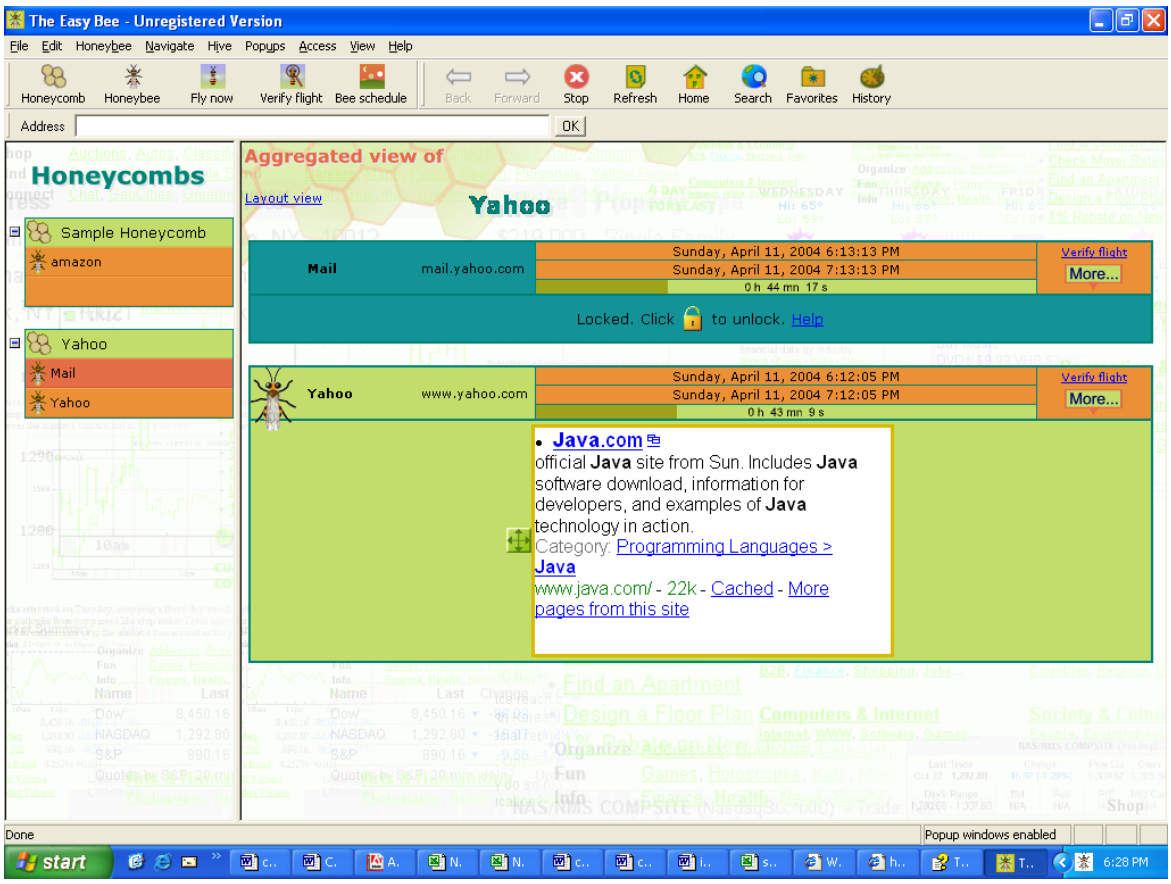


Figure 12: Password-protected Yahoo Mail is a "Private Bee"

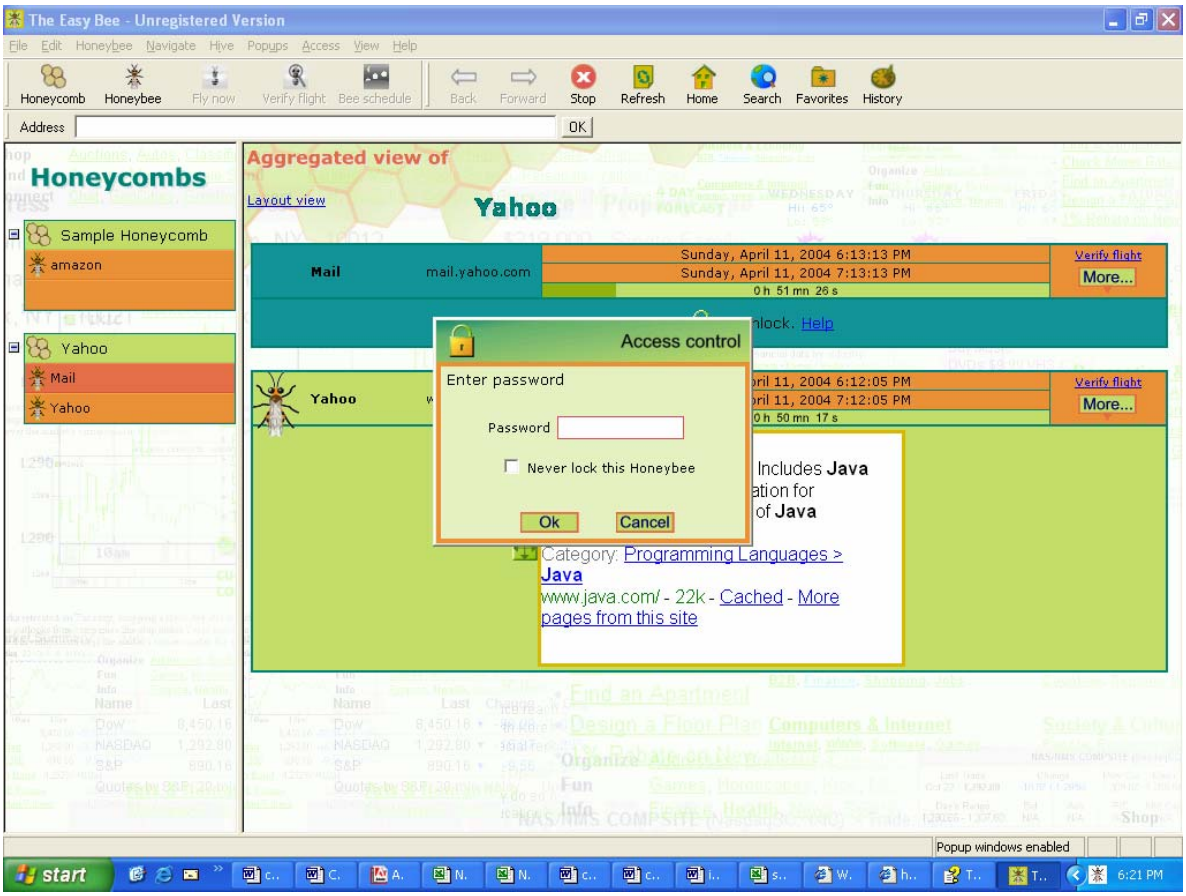


Figure 13: Unlocking a private bee requires entering a global password

4.11 Visual Web Task[27]

Visual Web Task provides a simple user interface that essentially functions as a WAD-oriented parser. It makes use of the underlying HTML tag hierarchy to create a set of extraction rules based on training examples provided directly by the user. As a commercial web wrapping tool, VWT comes with a host of features including a rather robust retrieval functional layer that is able to handle cookies, SSL and interpret scripts. However, it does not support HTML headers such as refreshes and redirection.

It is possible to construct a simple SQL interface using VWT, for web pages that support several attributes that can be altered. For example, a Yahoo home page contains several parameters for customization, allowing a user to specify the number of results per page or to choose which "national" Yahoo site to access. Using VWT, one can create a user interface that allows a user to query say, only the Yahoo Australian site and to limit the number of search results to 30 per page. A screen short of such an interface can be found below in Figure 14.

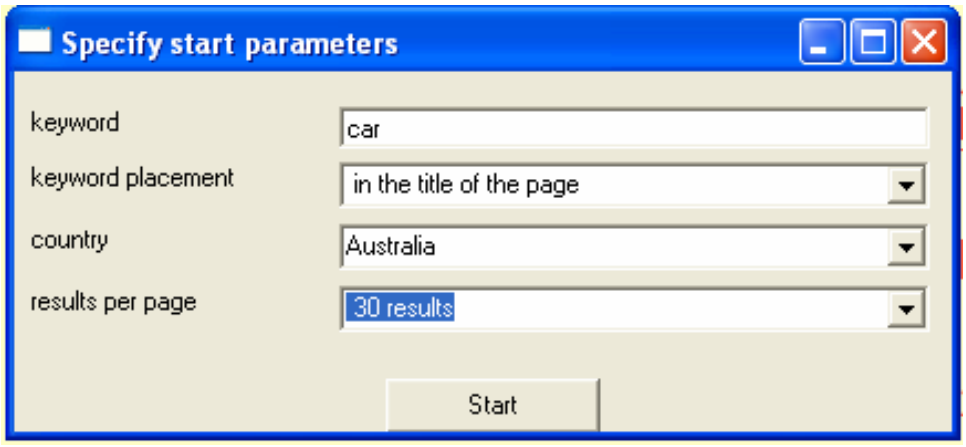


Figure 14: A simple VWT SQL interface.

Visual Web Task provides an easy-to-learn graphical environment that allows a user to capture WAD web documents with relative ease. It begins by asking a user to enter a new file name for the wrapper that VWT is about to generate. Then, the wizard leads the user to the navigation model where he is supposed to guide the wizard to a destination web page by supplying all the relevant information including a URL, username and passwords, if applicable, or any other actions that will lead to the eventual web page. A snapshot of this process is shown below in Figure 15.

Figure 15: The navigation model. VWT records a user filling in a required URL, and selects an item from the drop-down menu and fills in a text box.

After coming through the navigation model stage, VWT asks the user to select the information that needs to be extracted and displayed. Following on previous "JAVA books" example, the first item on the list (known as the ROW) is selected, leaving VWT to automatically generate a set of extraction rules that will ultimately capture the list of results. The left pane on Figure 16 shows the process in which each step is being captured. Users can also make use of the COLUMN section on the left pane to divide the row into columns using HTML tags, allowing say, the title of the book to be separated from the descriptions. If according to plan, this added feature allows the user to generate different columns in which one will display the title of the book and the other will contain the description of the book. It is found, however, that this feature is more effective in showcasing the text of the items captured on one side, and the underlying URL of the text on the other side.

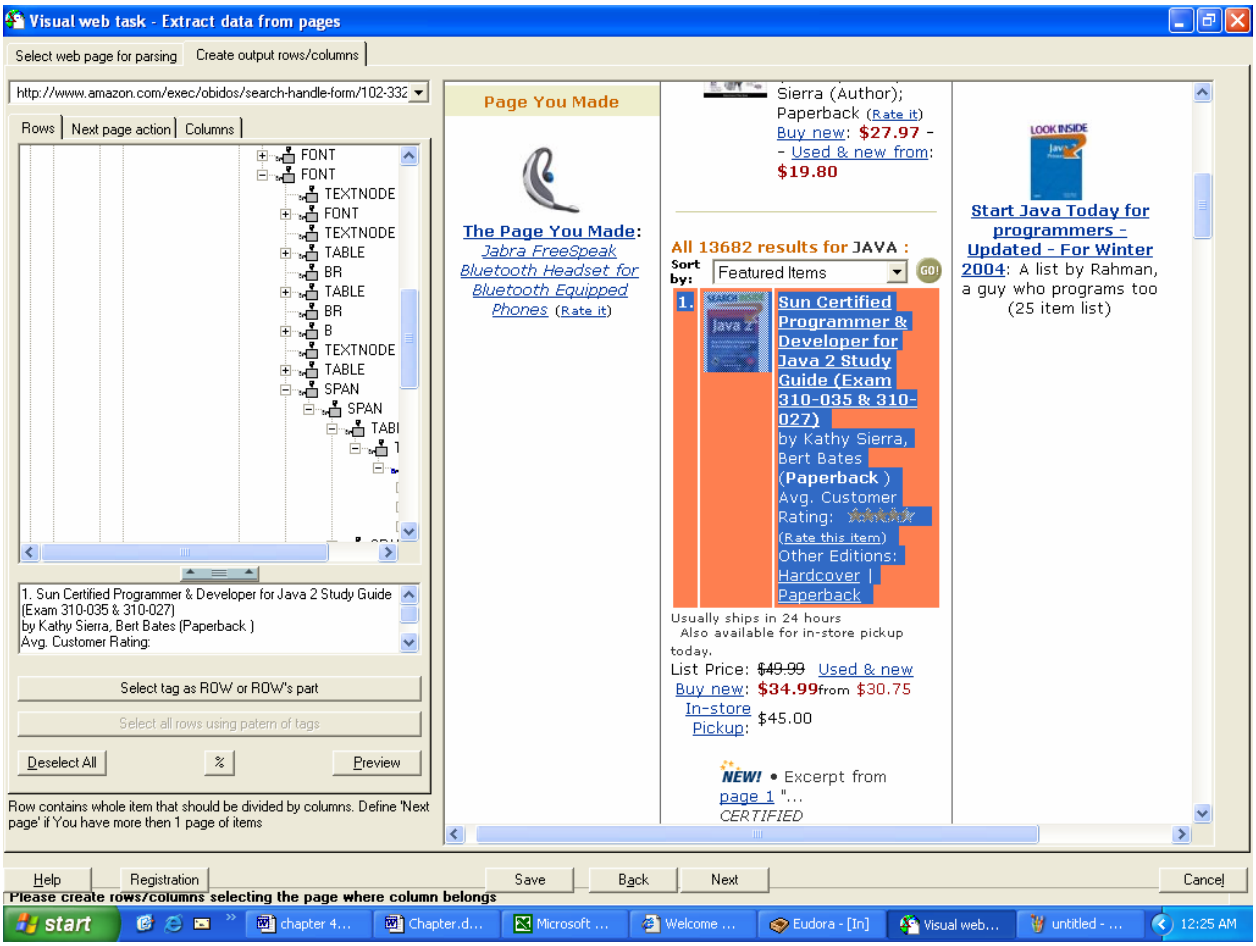


Figure 16: A user selects the first item from the list meant for extraction

Similarly, to activate VWT to capture the rest of the list on subsequent pages, users can click on the 'next page' or 'more results' buttons under the NEXT PAGE ACTION section. By clicking 'yes' to the pop-up dialog box, VWT will automatically generate the extraction rules for this action. This is shown in Figure 17.

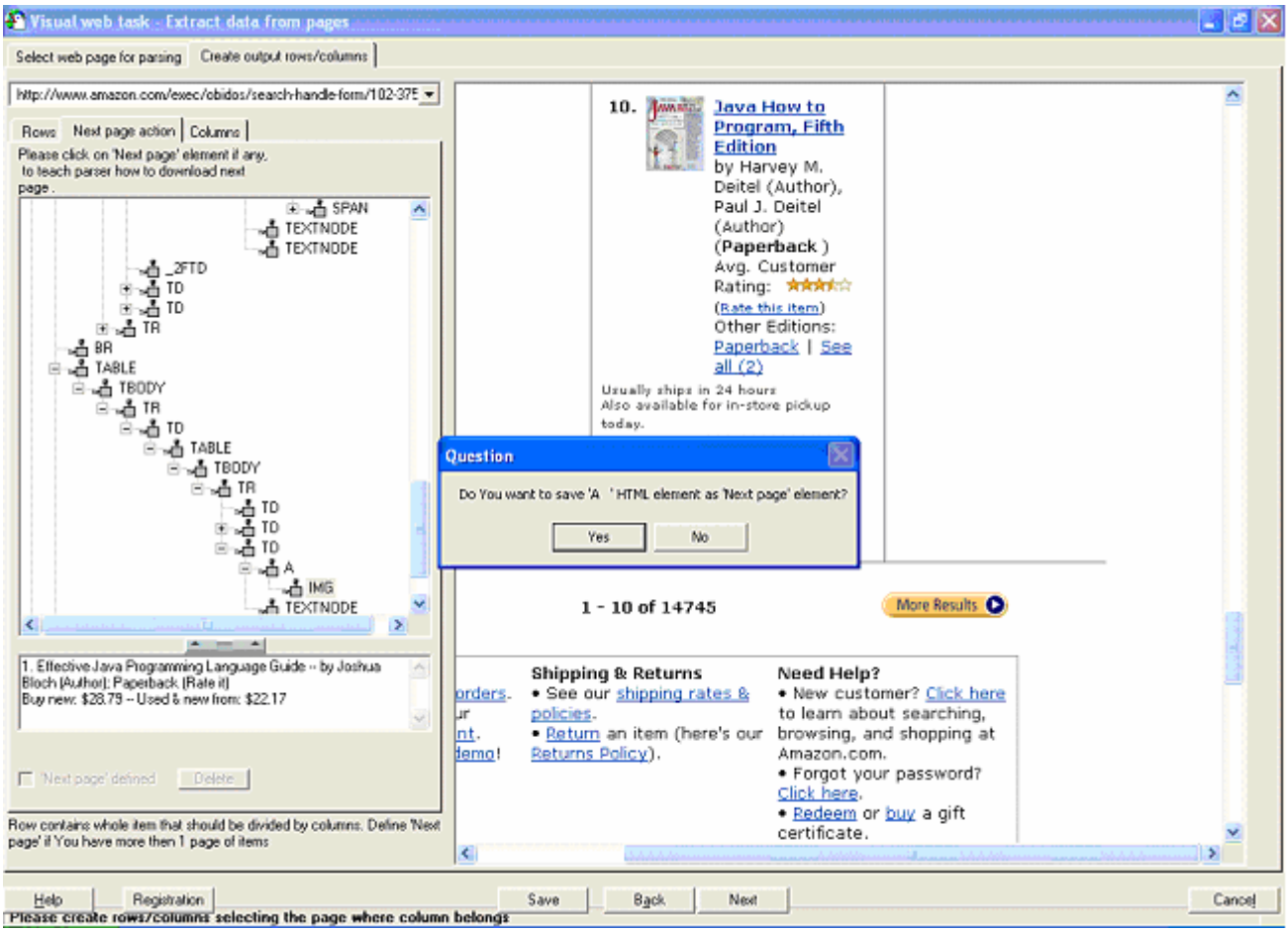


Figure 17: VWT asks user to extract from multiple pages

For output, VWT allows the user to choose between two modes: a comma delimited text/XML file or a database file. To this end, the wrapper generation process is mostly complete. With a push of a button, VWT will open up a new window and start to execute the wrapper program. Figure 18(a) below shows how the status of the wrapper in extracting web data whereas Figure 18(b) shows the preview results with the URL on one side and the title on the other.

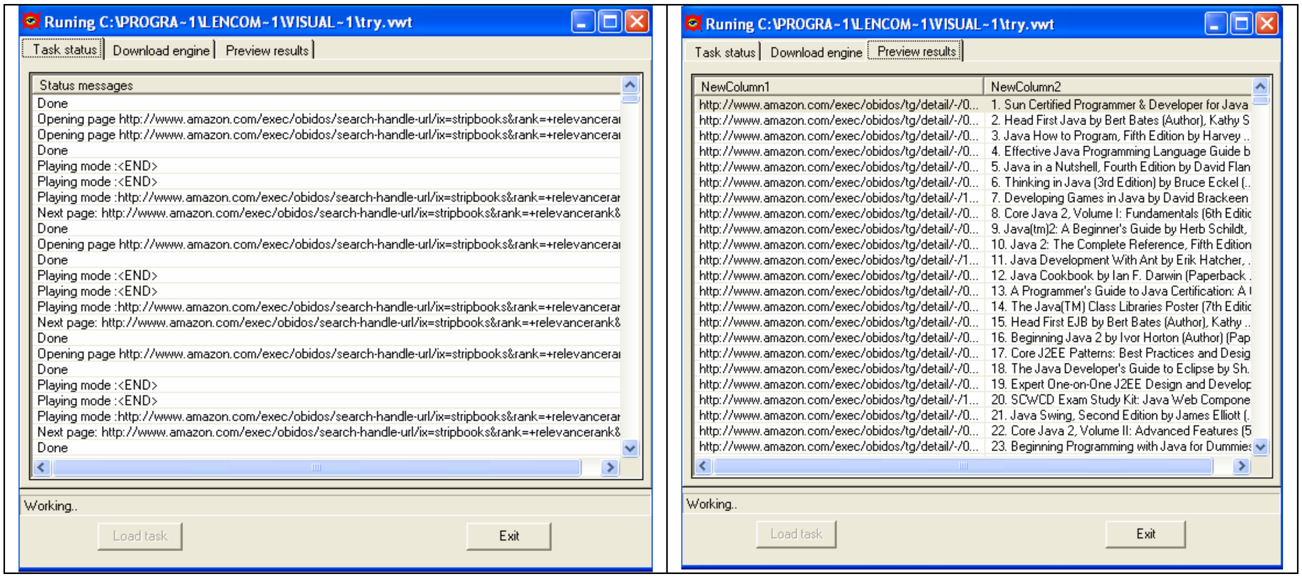


Figure 18 (a) & (b): (a) shows the task status of the extraction while (b) shows the preview result of URL and the corresponding text value.

4.12 Visual Wrapper[10]

Visual Wrapper is a software component of the extended family of Lixto Suite, and is responsible for navigating the world-wide-web for data extraction. Through a graphical user interface, a user feeds one or more training examples to Visual Wrapper which then effectively guide the toolkit into creating wrappers that mimic the actions/procedures of the training examples. In other words, Visual Wrapper records the actions of the training examples, and transforms these actions into web wrappers (written in its proprietary scripting language Elog) which ultimately are used for web data extraction purposes.

To understand how Visual Wrapper works, one needs to look at its underlying architecture. Figure 19 is a diagrammatic representation of how Visual Wrapper interacts with other component modules in its creation and execution of wrappers. Through the Lixto Visual Wrapper Designer, a set of matched instances are derived from a given set of sample web documents which contain pre-labeled patterns and pattern instances defined by the user. After that, the Designer translates this set of matched instances, otherwise known as extraction rules, into an Elog wrapper program. Once a wrapper is created, the Lixto Visual Wrapper Executor takes over and begins to execute this wrapper on other similarly structured web pages. The extracted data are then translated into XML and an XML companion of the respective web page(s) is generated.

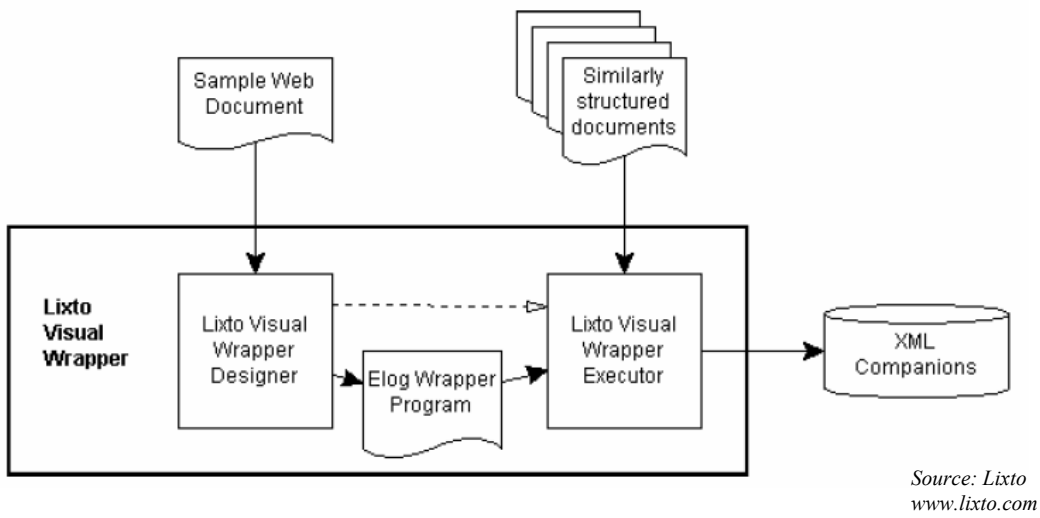


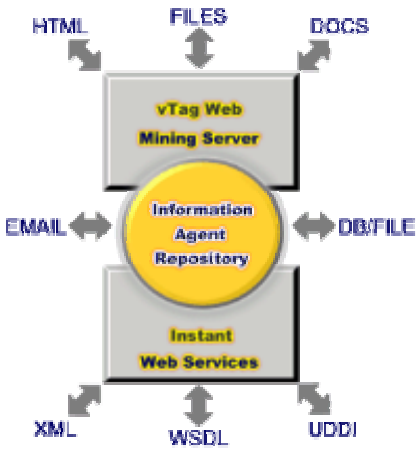
Figure 19: Interaction within and outside Visual Wrapper

Visual Wrapper is robust in terms of its retrieval capabilities. It is able to handle HTML headers, HTTP methods, cookies, SSL, authentication and even script interpretation. It supports web crawling, allowing the wrappers to go into many layer beyond the original starting page, given by the user. It has a SQL interface that effectively transforms the web into one giant database by allowing users to send queries directly. Apart from that, it also offers support for a range of APIs, ODBC and JDBC database drivers and an XML-based output format which can be transported into other formats for alternate uses.

4.13 vTag[28]

VTag is marketed as a web content mining and integration solution. It is able to extract data from the Internet, intranets and extranets, and has the ability to support more advanced retrieval features such as cookies, SSL and scripting languages. Its robust web crawling features allow a wrapper to log on automatically, fill in form and searches deeper into the web by following multiple links. Its adoption of XML also means vTag provides additional flexibility in terms of delivery options. In fact, a vTag's user is able to specify how the data is to be transmitted, whether it is through cell phone, hand-held devices or the more traditional PC applications.

This toolkit differentiates itself through its extensive use of XML-related services, and its range of delivery options. However, despite the seeming complicated offers, the underlying architecture is rather straightforward. Using an information agent repository which is essentially the web wrapping engine where the wrappers are stored and created, vTag sends out these wrappers to various locations, including HTML web pages, files, documents, email messages and databases to retrieve relevant data. Upon retrieving the data from the wrappers, the wrapping engine then converts the data into XML. From here, instant web services are enabled through a series of add-on features that include WSDL and UDDI which support the publication web services to the wider community on the Internet. The architecture can be succinctly summarized in Figure 20 below:



Source: Cannotate Technologies
www.cannotate.com

Figure 20: vTag's architecture

4.14 WebDataKit[29]

WebDataKit comprises of three core modules: a customized SQL for HTML and XML data sources, a customizable search engine kernel and a JAVA API known as WDBC, or WebDataBaseConnectivity (a dialect of JDBC). It does not support as many features as most other commercial web wrapping toolkits, and features such as support for SSL and cookies handling are believed to be absent. At the same time, its almost exclusive focus on data extraction from HTML-based web pages means WebDataKit is not ideal for integrating disparate data sources within internal file systems. By not offering regular expression or similar pattern matching algorithms, WebDataKit also limits itself to parsing web pages that are written based on the HTML tags.

One of the major extraction features that WebDataKit supports is the ability to query live web pages using modified SQL queries designed for querying HTML and XML pages. Its WDBC adapter allows a user to treat the web as a relational database as though he is querying database tables using JDBC, an API that provides database connectivity in a heterogeneous environment between different databases running on different platforms. The simplicity of this toolkit means that WebDataKit does not have an extensive mapping or mediation features for post-extraction activities. Its text-based only output also implies that WebDataKit is unable to offer a wide range of delivery options nor is it feasible for it to perform multiple data transformations.

4.15 Webinator[30]

Written in Texis's Web Script language called Vortex, Webinator is a web walking and indexing solution that affords a web site administrator with an interface to collect HTML and other documents. It consists primarily of three Vortex scripts and each of them serves a different function including an administrative interface, a site walker and indexer and lastly, a search function.

Webinator is a self-contained web wrapping toolkit with a comprehensive range of features supporting its retrieval, extraction and mapping functional layers. Apart from its support for SSL, cookies, scripts interpretation and HTML headers, Webinator also provides support for meta data and proxy servers, in addition to its ability to index many web sites into a single database. Equipped with an SQL interface, Webinator also allows a user to treat the web as a relational database for which he can send queries directly.

Perhaps the most unique feature of this toolkit, however, is Vortex's ability to support many pattern-matching formats. Among the many formats that Webinator supports include natural language, logic, regular expressions, quantities, fuzzy patterns, relevance rankings and proximity controls. This is perhaps not too surprising given that the wrappers created using Webinator have to be manually generated. Such manual writing of codes allow users to introduce custom features to the wrappers, allowing data extraction to perform according to a different set of extraction rules, and hence the support for different extraction techniques.

Webinator comes in 5 different versions, with the more expensive ones offering a greater selection of support features, including Java plug-ins and multiple document formats.

4.16 WebQL Studio[6]

A dialect of SQL, WebQL is a programming language that provides a set of SQL-like syntax to tackle data extraction and data integration problems, with a focus on the Internet. A flow diagram of WebQL studio that illustrates the high-level functions of this toolkit is provided below in Figure 21.

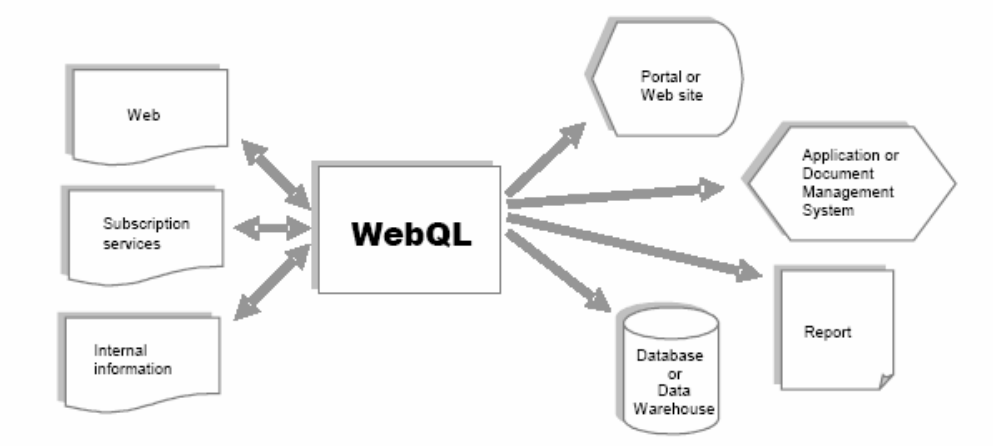


Figure 21: Flow Diagram for WebQL Studio

Source: QL2 Solutions www.ql2.com

As with any other manual web wrapping toolkit, WebQL studio is capable handling a huge variety of tasks, subject to the skills of the developers. For example, it has support for many retrieval features such as HTML headers, HTTP methods, cookies handling, authentication, SSL support and so on. Although there is no inherent feature that would allow WebQL to

automatically recognize and interpret script language found in some web pages, one can easily program WebQL to directly mimic the function of script, allowing a wrapper thus generated to function as if it is able to interpret script languages. Despite the fact that WebQL is a programming language in its own right, it employs extensive use of regular expressions to specify its extraction rules, making WebQL studio essentially a WAS toolkit.

Apart from providing SQL support, WebQL also has a web crawling ability that extends into navigating deep web links and filling forms. It is capable of accessing FTP sites and has the ability to schedule a wrapper for extraction at specific time and at regular intervals. In terms of its mapping layer, WebQL produces output in XML (and other formats such as .doc and .pdf). It also supports a rather large range of APIs that include COM, C++, JAVA, .NET and a few others, in addition to having a ODBC database driver.

To indicate how WebQL studio actually works, we provide a simple example for which this toolkit is asked to generate a wrapper that crawls through Google to display URLs associated with the submitted query "JAVA". Figure 22 below shows the actual line of codes needed to type in by a user.

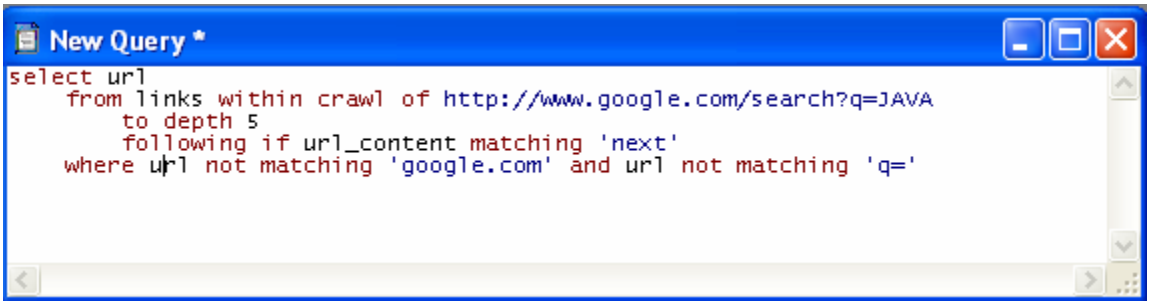


Figure 22: WebQL codes

WebQL studio provides a generous range of tools that aid in the execution of wrappers like this. By pressing the "run" button, WebQL studio takes the user to the next window that display the run-time environment including the list of URL captured. There is also a section that outlines the activities undertaken by the wrapper. Both of these events can be seen as captured in the screen shot below. Note that WebQL only returns 5 pages of the Google search result because it is indicated that the depth of search is to be limited to only 5 levels, as shown in Figure 23 on the following page.

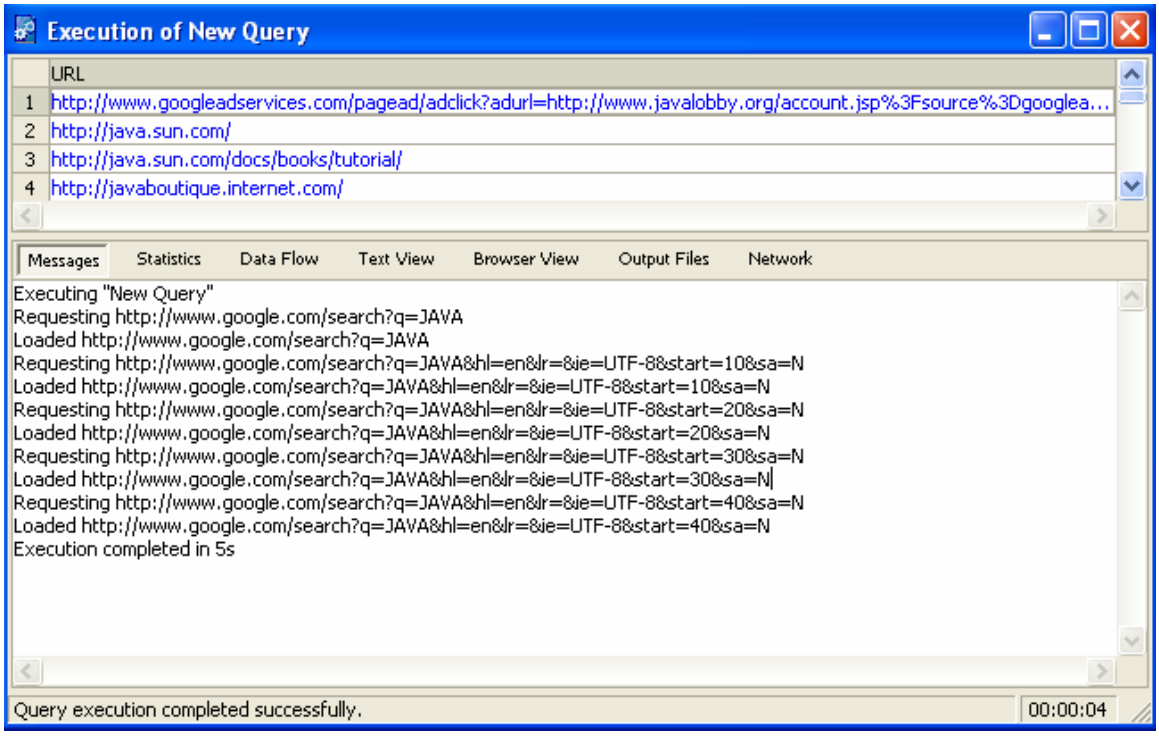


Figure 23: Execution of a WebQL code

4.17 WIEN[13]

Developed by University College of Dublin in Ireland, WIEN generates wrappers automatically using machine-learning algorithms. It is targeted at web pages that contain organized data structure, for example, a list of results returned by a search engine. Often, such a web page contains lists of items organized in a systematic order, allowing a user to easily mark the text on the web page according to his requirement. These marked texts are then used as training example to feed into another program called HLRT wrapper inductor that, like most wrapper induction tools, generates delimiter-based extraction rules based on the examples provided. The extraction rules are then refined further as each new training example is added. Subsequent web pages that need to be extracted can then make use of an existing set of extraction rules provided these web pages have similar document structure as the training examples.

Given the current state of artificial intelligence, it is no surprise to find that web wrapping engines that make use of machine learning algorithm to automatically generate web wrappers are rather limited in terms of the types of web pages that they can wrap and the amount of information that they can be expected to extract. As such, WIEN is short of the many features that are available in other web wrapping tools including SSL and script interpretation ability. This is due partly to the fact that such features usually do not exist in simple WAD-type HTML web pages. Apart from these apparent limitations, there is also no SQL support for WIEN.

Coupled with the lack of web crawling ability and the lack of API, WIEN's functions are limited to parsing multiple web pages that share the same structure as its training sets. The text output that it generates also means it cannot be manipulated to other format as easily as XML, limiting its ability to usefully mediate the extracted data.

4.18 WinTask[31]

The range of tasks that WinTask is capable of extends beyond web data extraction. However, for the purpose of this thesis, we shall focus on its web extraction features.
WinTask provides a user friendly interface that makes the automation of mundane web data extraction. Through a recorder, user of this toolkit has to provide a training example whenever he wants to generate a wrapper to perform a specific task. For example, if a user wants to automate the extraction of his email account, he needs to supply the username and password of his account, and performs how he retrieves his email manually. Then, his actions would be captured and replayed by a wrapper whenever this wrapper is asked to run. The range of activities that the user can undertake is pretty big, and includes most of the functions that a normal web browser can support, such as to fill in forms, deep web link, authenticate users and download files.

WinTask uses a scripting language that is similar to Visual Basic, known as WinTask Programming Language (WPL). However, users of this toolkit does not need to know how to develop using this scripting language as most of the actions can be captured using the recorder, which would then translate the actions into line-by-line codes. In the rare event that a user needs to code a wrapper by hand, coding in WPL proves to be rather easy to pick up, with approximately slightly over 100 syntaxes for use. However, most of the time, users only have to become familiar with a few of them, such as WriteHTML, SelectHTMLItem and ClickHTMLElement. Figure 24 shows the codes behind the wrappers generated using WinTask, which is tasked to display a list of JAVA related books from the Amazon.com web site.

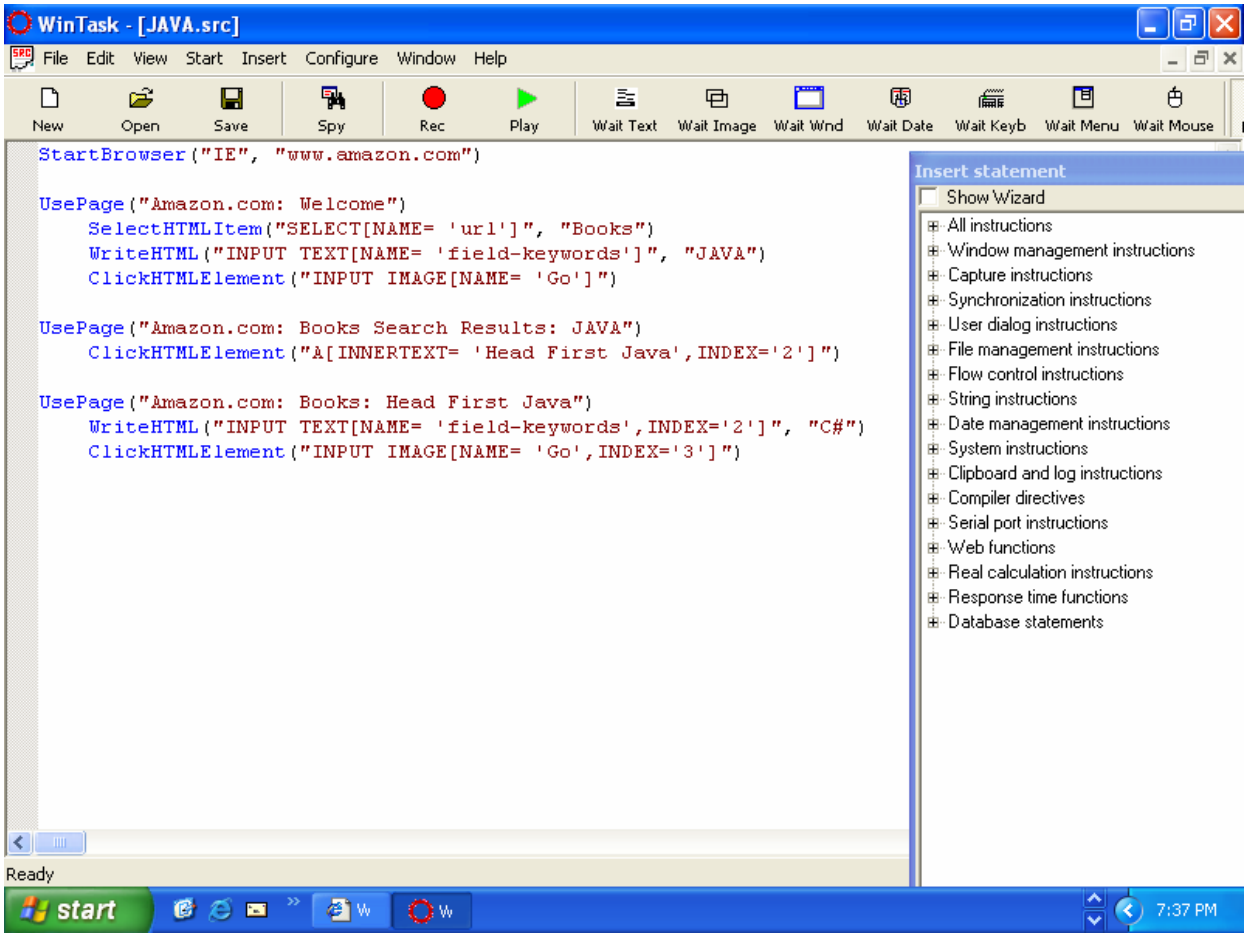


Figure 24: A Piece of code written using WinTask Programming Language

In terms of its retrieval layer, WinTask handles most of the common features, with support extended to SSL, cookies handling and HTML headers. However, as with most of the web wrapping technologies in its range, WinTask does not have the capability to interpret script language such as Javascript. This is most notably demonstrated by its inability to wrap an Expedia.com web site properly. Figure 25 shows the error message that we encountered when we tried to wrap an Expedia web site using WinTask.

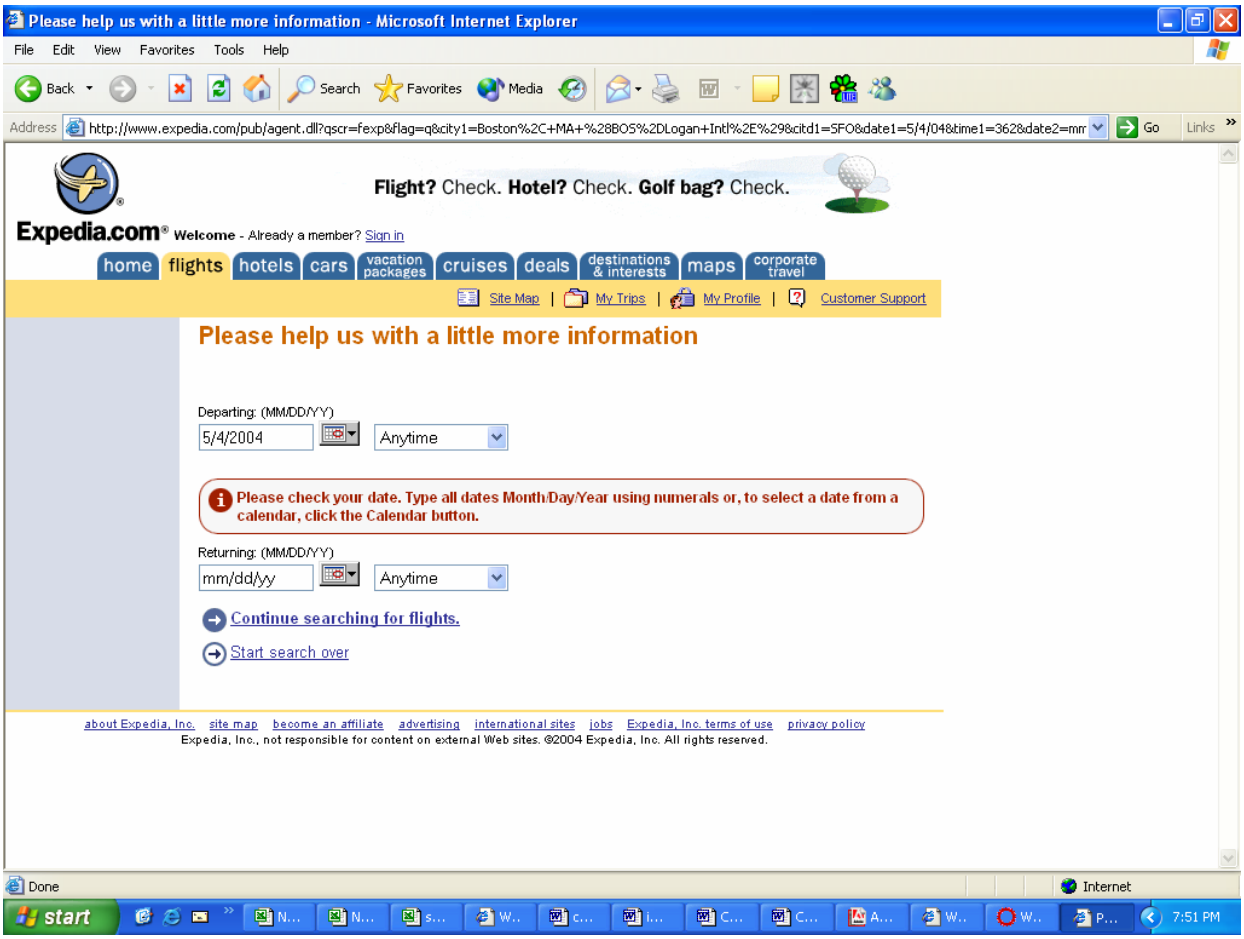


Figure 25: Error Message for failing to interpret script on Expedia website

In many ways, WinTask is perhaps very similar to The Easy Bee that was mentioned earlier on. Like The Easy Bee, WinTask does not come with an SQL interface that would allow a user to query the web directly. Instead, individual wrappers have to be generated for each possible query "action", although the trouble of doing that is negated by the ease of wrapper generation. Similarly, it follows The Easy Bee format by only allowing for text or HTML-based output formats. However, unlike The Easy Bee, WinTask offers support for FTP and is able to download files through its wrappers. It also comes bundled with a ODBC database driver useful for connecting with other software packages, in addition to its API support that include a COM interface.

4.19 XWRAP Elite[9]

XWRAP Elite is a web-based wrapper application that specializes in extracting data objects from WAD web pages. According to XWRAP own lingua, this toolkit works best on web sources which are "data objects rich", or one where the objects/elements follow a certain structure. A typical example is the search result page generated through a search engine such as Google or

Yahoo. XWRAP Elite generates wrappers as JAVA classes and transforms HTML document into XML format.

The extraction algorithm of this toolkit relies heavily on heuristics methods, and hence the accuracy of extraction increases as the number of objects goes up. As such, XWRAP Elite is particularly useful in wrapping search results. On the other hand, precisely because of the emphasis on heuristics, this toolkit is not ideal for unstructured web pages where repetitive data objects are usually a lot more limited. Unlike most commercial tools, XWRAP Elite does not use a comprehensive web client for its retrieval layer, and is therefore ill-equipped to handles such features as cookies and SSL, in addition to interpreting script languages. It also does not offer an option for scheduled extraction, making this tool unsuitable for repetitive extraction activities such as keeping track of a stock quote.

The next two figures illustrate how XWRAP Elite works. Figure 26 below shows the web page that is to be captured whereas Figure 27 on the following page illustrates the way in which XWRAP Elite makes use of the underlying HTML hierarchy to parse the results from this Yahoo search into a document tree.

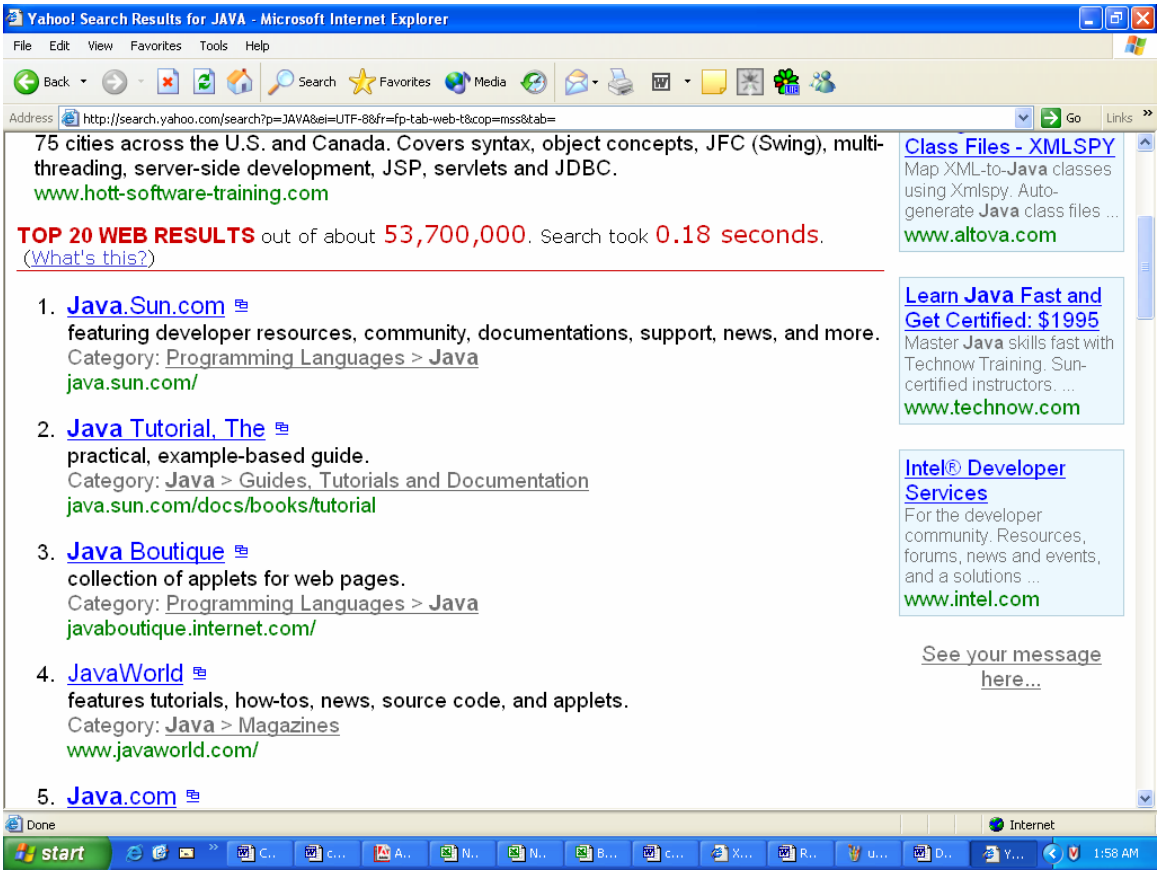


Figure 26: The "source" web page that will be captured by XWRAP Elite

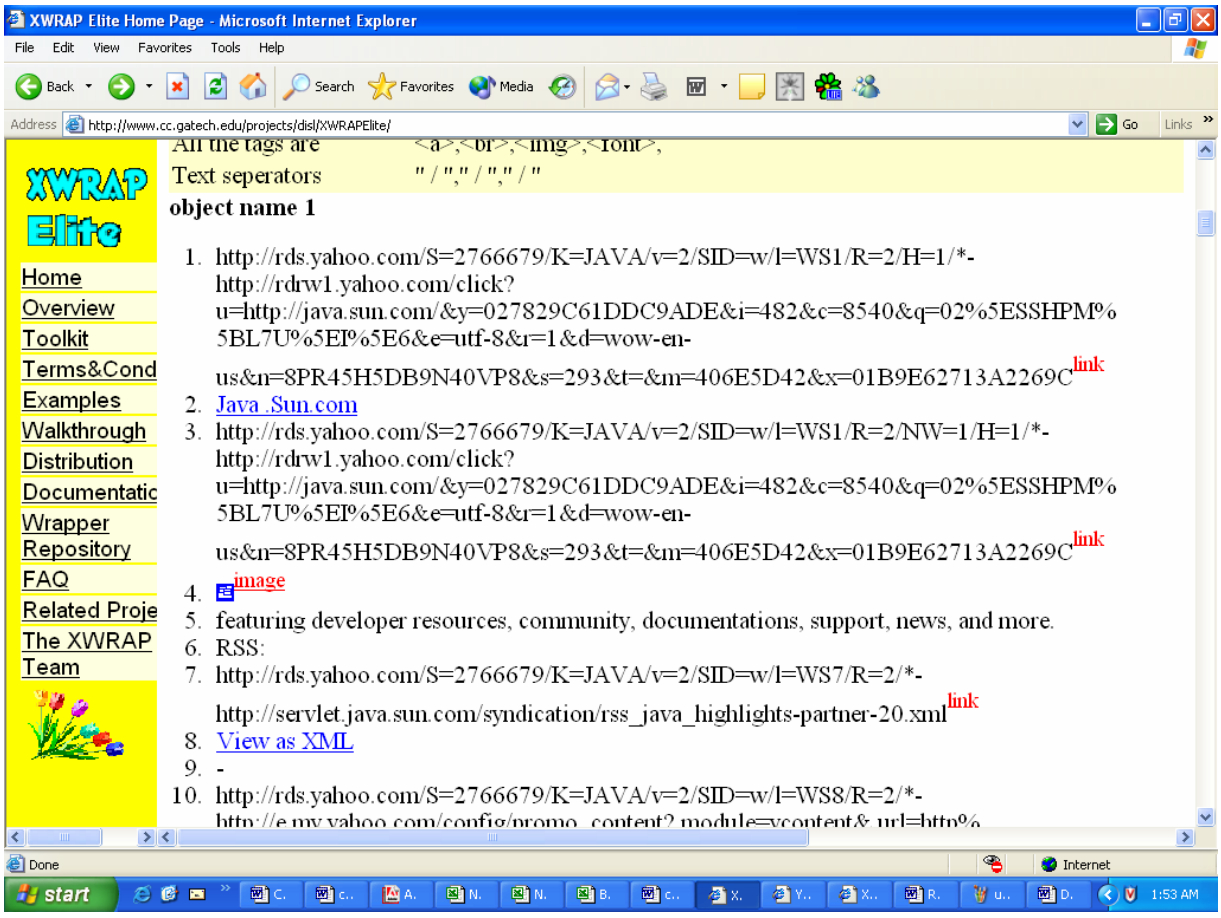


Figure 27: XWRAP Elite uses a WAD approach

Despite its relatively disappointing retrieval functions, XWRAP Elite does bring out a certain number of advanced features, including support of a query interface that makes use of XML-Wrapper Query Language (XML-WQL) for accepting application query requests. More importantly, it is one of the most user-friendly tools that are being tested in this survey. The online walkthrough example provides enough guidance to build a wrapper in 9 simple steps, as shown below at the top of the screen shot in Figure 28.

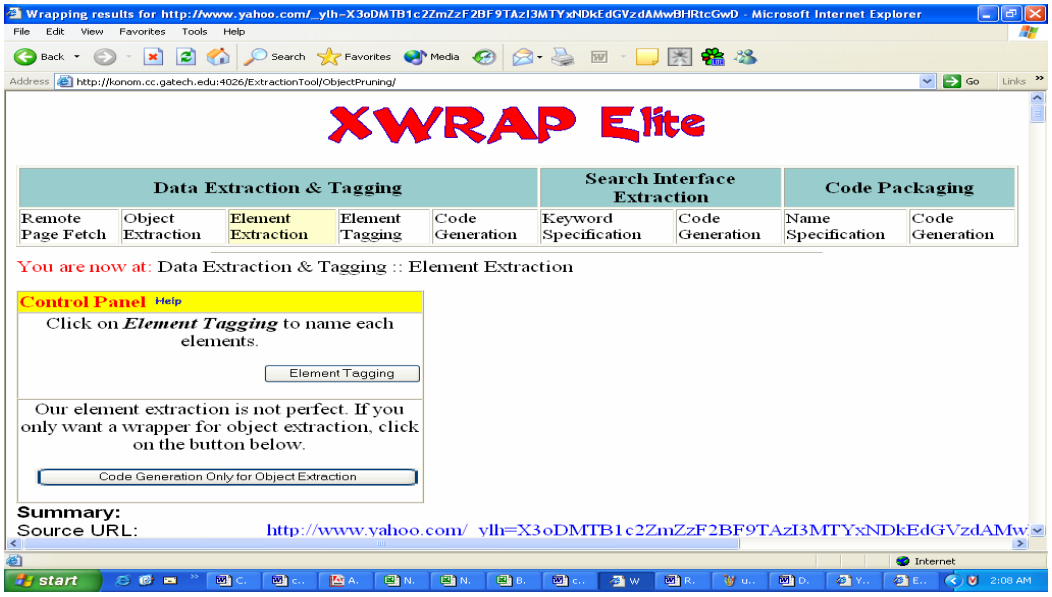


Figure 28: Screen shot showing XWRAP Elite can wrap a web page in 9 simple steps, producing 3 JAVA codes in the process.

4. 20 XRover[32]

The XRover suite consists of two core modules, the XRover Agent Manager that allows a user to manage and execute agent tasks; and the XRover Site Mapper that is used to create agents (or web wrappers) to parse the web for information.

In effect, the XRover Agent Manager is the primary interface between a user and the world-wide-web as it is used to coordinate the execution of the wrappers including such functions as add, delete and scheduling. It can also be used for other mapping functions such as specifying how the extracted data are to be stored and the input and output formats of the wrappers. The Agent Manager is also responsible for the retrieval layer of the toolkit including such functions as support for cookies, HTTP and HTTPS protocols, authentication, redirection handling and script interpretation. Through the Agent Manager, a user also enjoys a certain degree of freedom in customizing the output of the data. For example, he can adjust the parameters to strip out the accompanying HTML tags and remove white space, from the raw data extracted from other web sources. With a ODBC driver, Agent Manager also affords the user to write to other ODBC-compliant software such as Excel, on top of the XML files that it normally generates. The Agent Manager also has a built-in web crawling capability that will allow the wrappers to navigate the world-wide-wide through a single URL, on a on-demand or scheduled basis.

On the other hand, XRover Site Mapper is an agent or wrapper generating device. It is designed to allow a user to generate wrappers using a set of training examples. Like most other semi-automatic wrapper generation toolkits, XRover comes bundled with an editor that can be used for hand-coding the wrappers for cases whereby the targeted web sites are too complex. To use

this editor, however, a user needs to be proficient with regular expressions and the Prolog language, a scripting language developed by XSB.

Chapter 5: Application of the 2-Tier Taxonomy – A Case Study

The aim of the 2-tier taxonomy system is to enable a manager, a developer or anyone who needs to find a suitable web wrapping toolkit to do so in an organized and systematic manner. For instance, by using the 2nd-tier taxonomy, a beginner looking for a simple toolkit would probably be inclined to check out some of the semi-automatic toolkits and avoid the manual ones. Similarly, an expert programmer looking for a solution to create highly customized wrappers for complex web pages may want to spend more time evaluating one of the ten toolkits with a "12" rating.

While it is possible to use either the 1st-tier or the 2nd-tier taxonomy independently, a more desirable approach is to first use the 2nd-tier taxonomy as a filter before zooming in to the more detailed 1st-tier taxonomy. To illustrate how this is done, consider this hypothetical question: "What is the best substitute for Cameleon?"

5.1 Step One: How to Use the 2nd-Tier Taxonomy

As briefly described above, the 2nd-tier taxonomy can be used independently or it can be used as a filter for answering questions such as "What is the best substitute for Cameleon?" where more detailed analysis is often needed. In this case, the 2nd-tier taxonomy provides a high-level summary of the characteristics of Cameleon. By reading it, a reader is able to immediately gain a basic understanding of the capabilities of Cameleon. For example, he will know that Cameleon offers SQL support – a "joins" operation between multiple web sites is possible under Cameleon. He will also know Cameleon contains at least one database driver so database connectivity is not a concern, despite the fact that he has yet to know what types of database drivers are supported.

The most useful feature of the 2nd-tier taxonomy, however, lies in its ability to allow users to compare a number of web wrapping toolkits. In this case, a user of this taxonomy will quickly discover that there are 9 other comparable toolkits to Cameleon, as these 9 toolkits support the same features as Cameleon. However, only 5 of the 9 toolkits use the same extraction method as Cameleon. This can be derived by looking at the HTML-awareness row in the taxonomy. Cameleon treats web pages as a sequence of characters and therefore does not rely on the underlying HTML hierarchy to parse web pages into a document tree, and hence is not HTML-aware. Similarly, Integration Platform, NQL, vTag, WebQL and Robosuite are also not HTML-aware. These 5 toolkits are therefore selected as potential candidates to substitute Cameleon. On the other hand, the remaining toolkits like AgentBuilder, ContentMaster, Visual Wrapper and Webinator all make use of the HTML tags for data extraction and are therefore excluded. Table 3 on the following page is a duplicate of the 2nd-tier taxonomy showing the selected toolkits in a darker shade.

		Agent Builder	Cameleon	ContentMaster	DB2 IIC	Integration Platform	Mine the Web	NQL	RoboSuite	SearchExtract	The Easy Bee	Visual Web Task	Visual Wrapper	vTag	WebDataKit	Webinator	WebQL	WMEN	WinTask	XWrap Elite	XRover
1	Degree of Automation	S	M	S	S	S	M	S	S	S	S	S	S	S	M	M	M	A	S	S	S
2	HTML Awareness																				
3	HTTP Methods																				
4	HTML Headers																				
5	SSL Support																				
6	Cookies Handling																				
7	Script Interpretation																				
8	SQL Support																				
9	Web Crawling																				
10	Scheduled Extraction																				
11	On-demand Extraction																				
12	Ouput Format (XML)																				
13	API																				
14	Database Driver																				
		12	12	12	11	12	6	12	12	8	9	9	12	12	6	11	12	4	9	7	12

Table 3: The 5 selected toolkits and Cameleon are darkened.

It is also tempting to conclude just from Table 3 that WebQL should be the most ideal candidate to replace Cameleon since both toolkits rely on manual generation of web wrappers. To allow for meaningful comparison, however, all 5 toolkits will be used in the next round of examination using the 1st-tier taxonomy.

5.2 Step Two: How to Use the 1st-Tier Taxonomy

The next logical step in answering the question "What is the best substitute for Cameleon?" is to examine at deeper level of granularity each of the 5 selected toolkits. This is done by examining their respective features and functionalities using the 1st-tier taxonomy. For ease of readership, the original 1st-tier taxonomy is reproduced showing only the 5 selected toolkits and Cameleon in Table 4 on the following page.

Since 14 out of the 21 features/functionalities are already included in the 2nd-tier taxonomy, comparison at this level will therefore focus on the remaining features, including the platform supported, GUI , Editor, FTP support, input format, NLP support and types of scripting languages. Using a simple Process-of-Elimination (POE), Integration Platform and vTag are deemed inappropriate substitutes for Cameleon because of their lack of support for editor and scripting languages, and hence their inability to generate web wrappers manually. As a result,

only 3 toolkits: NQL, RoboSuite and WebQL are left as potential candidates to replace Cameleon.

	Company	MIT Sloan	WebMethods	NQL Tech	Kapow Tech	Cannotate	QL2 Software
	Tool	**Cameleon**	**Integration Platform**	**NQL**	**RoboSuite**	**vTag**	**WebQL**
	General						
1	Degree of Automation	Manual	Semi	Semi	Semi	Semi	Manual
2	HTML-Awareness	no	(no)	no	no	no	no
3	Platform	Windows	(Windows, Unix)	Windows	Windows, Linux	web-based	Windows, Solaris, Linux
4	GUI	(yes)	yes	yes	yes	yes	yes
5	Editor	yes	(no)	yes	yes	no	yes
	Retrieval Features						
6	HTML Headers	yes	yes	yes	yes	yes	yes
7	HTTP Methods	yes	yes	yes	yes	yes	yes
8	Cookies Handling	yes	(yes)	yes	yes	yes	yes
9	SSL Support	yes	(yes)	yes	yes	yes	yes
10	Script Interpretation	yes	(yes)	yes	yes	yes	(yes)
11	FTP Support	yes	yes	yes	(yes)	(yes)	yes
12	Input Format	text & binary	text & binary	text & binary	text & binary	text & binary	text & binary
	Extraction Features						
13	SQL Support	yes	yes	yes	yes	yes	yes
14	Web Crawling	yes	yes	yes	yes	yes	yes
15	Scheduled Extraction	yes	yes	yes	yes	yes	yes
16	On-deamand Extraction	yes	yes	yes	yes	yes	yes
17	NLP Support	no	no	yes	no	no	(no)
18	Scripting Language	regular expression		NQL	regular expression		WebQL
	Conversion Features						
19	Output Format	XML	XML	XML	XML	XML	XML
20	API	JAVA, .Net	yes	yes	JAVA, .Net	yes	COM, JAVA,.NET, C++
21	Database Driver	(ODBC)	ODBC, JDBC	ODBC, JDBC	ODBC	(ODBC)	ODBC

Table 4: A replicate of the 1st-tier taxonomy showing the 5 selected toolkits and Cameleon

5.3 Step 3: Analysis of Final Candidates

At this stage, the 3 final candidates, NQL, RoboSuite and WebQL are deemed to be comparable if not equivalent to Cameleon in terms of features and functionalities. As such, the next step is to examine the other features of each of these 3 toolkits that are not covered by the taxonomy. This

is achieved by analyzing the toolkits in greater details. This step is necessary to ensure the eventual "winning" candidate does not contain other limitations that might compromise its features relative to Cameleon. In the following section, an analysis of NQL, RoboSuite and WebQL will be presented.

5.3.1 NQL

Although classified as a semi-automatic web wrapping toolkit because of the presence of a browser recorder that allows wrappers to be generated using training examples, NQL was in fact developed as a manual toolkit based on the Network Query Language, also abbreviated to NQL. The Network Query Language is a dialect of SQL and is used primarily to create connected applications such as bots, intelligent agents, middleware and web applications. It can therefore be considered as a kind of shorthand for Internet and network programming.

The most obvious difference between NQL and Cameleon is the artificial intelligence capabilities that are embedded into the NQL toolkit. Using neural networking, Bayesian inference, pattern matching and fuzzy logic, a NQL toolkit allows users to extract data from free text, or documents that contain no apparent structure such as email messages. For instance, NQL supports a number of neural networks including Adaline networks, back propagation network, Kohonen networks and Bi-directional associative memory (BAM) networks. As an illustration, the Adaline network, shown in Figure 29 (which in turns shows 5 input nodes and 1 output nodes) below is trained using training examples in which input and output data sets are fed into the network repeatedly until it adjusts the weights between nodes to reduce the error rate and ultimately becomes a generalized system capable of processing new input values.

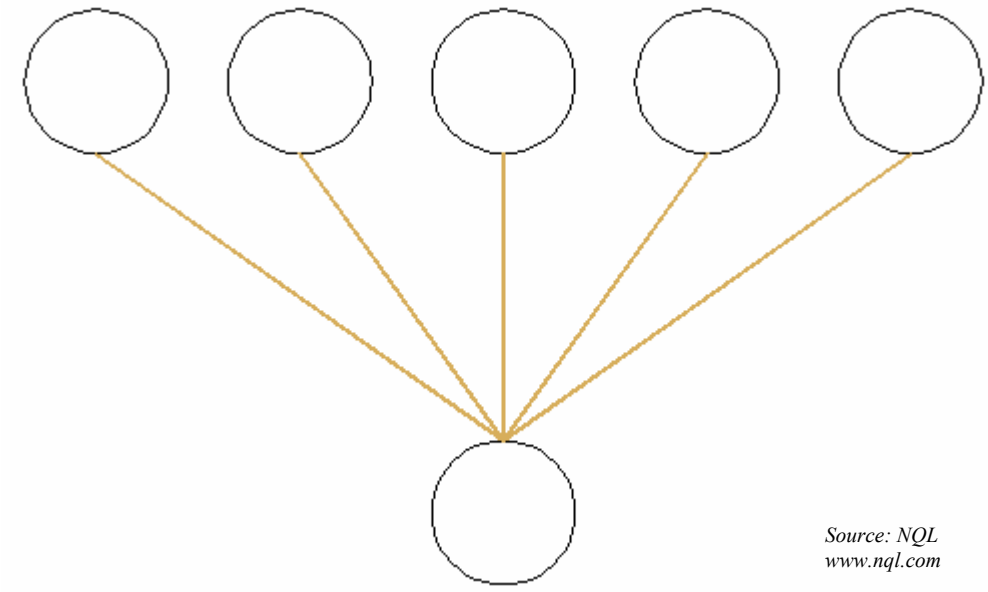


Figure 29: Adeline network with 5 nodes

At the same time, it is interesting to note that regular expression is used as part of the pattern matching package within NQL. This is similar to Cameleon, which uses regular expressions as part of its scripting language package. Therefore, it is possible for a user experienced in using regular expressions in Cameleon to migrate to the NQL environment with relatively few difficulties. In this regard, Cameleon acts as a subset of NQL. It is therefore reasonable to conclude that NQL is a suitable substitute for Cameleon based on the similarity of the scripting language, the degree of automation and other attributes described in the 2nd-tier taxonomy.

Verdict: Accepted

5.3.2. RoboSuite

According to Table 4, there is no apparent difference between Cameleon and RoboSuite, apart from the degree of automation of wrapper generation. Although RoboSuite is considered as a semi-automatic web wrapping toolkit, like NQL, it also comes with an editor where wrappers can be manually generated. However, as described in section 4.8.1, RoboSuite has a rather different underlying architecture compared to most other web wrapping toolkits, including Cameleon. This might pose a problem to the users already familiar with Cameleon's architecture and make the conversion to RoboSuite more problematic.

RoboSuite is made up of several application modules such as ModelMaker, RoboMaker, RoboServer and RoboManager. Each of these modules has a specific function and only two modules, ModelMaker and RoboMaker are involved in wrapper generation. ModelMaker, by definition, is "an application for modeling the objects used by a robot, such as the objects that the robot extracts and the objects that it accepts as input objects." It is also used to create domain models which in turn are used to define one or more objects. For example, a domain model designed to model airline information will define objects that are relevant to this information such as ticket prices and names of airports. On the other hand, RoboMaker provides a graphical user interface for users to create and debug wrapper in a semi-automatic way. While an editor is also provided for users to code the wrapper manually, this editor is part of the RoboMaker. This means a user is only able to make incremental changes to a wrapper created using training examples. In other words, unlike NQL, there is no separate editor where users can manually code a wrapper from beginning to end.

The biggest problem for using RoboSuite as a replacement for Cameleon therefore lies in the architectural difference. Users of Cameleon are not exposed to the domain model concept so migrating from Cameleon to Robosuite might not be hassle-free, even if both toolkits utilize regular expressions as part of their scripting language packages. RoboSuite's semi-automatic wrapper generation method also makes it a less ideal candidate to replace Cameleon compared to NQL.

Verdict: Rejected

5.3.3: WebQL Studio

WebQL Studio is a toolkit built around a proprietary scripting language called WebQL. Essentially, WebQL is a dialect of SQL and is targeted towards data extraction on the web. Someone who is familiar with SQL, such as a Cameleon user, will be able to easily switch over to WebQL with little training. For example, assuming a user wants to retrieve all the data in a database table using SQL, the query will be:

```
select * from TABLE
```

where TABLE refers to the database table from which the data is to be retrived.

In WebQL, the user can enter the following query if he is retrieving data from a file system:

```
select * from TABLE@DATABASE
```

where DATABASE refers to the database which contains the table.

On the other hand, if he wishes to directly query a web site, the query in WebQL becomes:

```
Select * from http://www.aaa.com
```

where the website http://www.aaa.com is any web site that he wishes to extract data from.

Of course, there are more noticeable difference that the one shown above. For example, consider the following SQL query (adapted from QL2):

```
select TABLE1.ID, TABLE1.XXX, TABLE2.XXX
        from TABLE1, TABLE2
        where TABLE1.ID=TABLE2.ID
```

In WebQL, the same query is written as:

```
select as TABLE1 *
        From TABLE1@DATABASE
join
select  *
        from TABLE2@DATABASE
        where TABLE1.ID=TABLE2.ID
```

Like NQL and Cameleon, WebQL also makes extensive use of regular expressions in its scripting language package. This means someone who is familiar in using regular expressions to code a wrapper in Cameleon will have a relatively less steep learning curve when he migrates to the WebQL environment.

As mentioned before, WebQL studio is the only toolkit that matches Cameleon exactly in Step One. This means that unlike NQL, WebQL studio does not come bundled with a browser recorder where a wrapper can be semi-automatically generated by feeding it with training examples. Instead, the effective use of WebQL studio depends largely on the skill level of the developers or users of this toolkit, who might be familiar with the WebQL scripting language and regular expressions. Both toolkits also support a range of features such as web crawling, SSL support, script interpretation and output format.

Although WebQL studio and Cameleon are identical in many aspects, subtle differences do exist. For example, WebQL studio supports a wider range of APIs compared to Cameleon. WebQL studio also comes with a number of "value-added" features such as anonymization which allows a web wrapper written in WebQL to hide its identity. This feature works in tandem with the WebQL Identity Protection Services. This service works by routing the page requests (or wrappers) sent by a user through a third party hosts, thereby protecting the identity of the requester. Other features in WebQL studio includes automatic page request throttling, error trapping and reporting, and graphical data-flow monitor.

From the above discussion, WebQL studio is considered as an ideal substitute for Cameleon. This is based on the similarities of features and functionalities between the two toolkits. Unlike NQL which is a superset for Cameleon, WebQL is more like a "peer", albeit one that is equipped with more "peripheral" features.

Verdict: Accepted

5.4 Conclusion

The 2-tier taxonomy provides a quick and effective way to help anyone to choose an appropriate web wrapping toolkit. In answering the question "What is the best substitute for Cameleon?" the 2nd-tier taxonomy is used to filter out a large number of incomparable toolkits quickly, while the 1st-tier taxonomy is deployed to extract only the most likely candidates. From the above discussion, out of the 3 toolkits that are selected from the 2-tier taxonomy, 2 of them are eventually considered as ideal substitutes to Cameleon. While not exactly perfect, it demonstrates the effectiveness of the 2-tier taxonomy to evaluate different web wrapping toolkits.

In addition, the eventual "winners", NQL and WebQL show a close resemblance to Cameleon not only in terms of features and functionalities, but also in terms of scripting language structure. The later properties qualify NQL and WebQL to be the "best substitutes" to Cameleon, from mere "substitutes".

Chapter 6: Policy Analysis of Database Protection

One of the key benefits of the emergence of web wrapping technologies is the integration of information from disparate sources, including the world-wide-web. Web wrapping technologies bring together information in a consolidated and coherent manner from web sites or other data sources which otherwise exist individually and independently from one another. By linking these disparate data sources together, web wrapping technologies effectively create a giant relational "virtual database" from the world-wide-web, allowing users to send queries directly to the web.

However, such a facilitation to extract data more efficiently and effectively from various web or non-web data sources creates a problem in terms of the ownership of this "virtual database". Many web sites which welcome manual browsing by ordinary consumers tend to reject the notion of having spiders or robots (i.e. wrappers) preying on their web servers. Companies grow uncomfortable with computerized wrappers that extract data from their digitized databases systematically and periodically, especially when these databases are compiled with much investment in terms of finances and efforts. As a result, there have been a plethora of court cases in recent years challenging the legality of data extraction through the use of web wrappers. The more well-known cases include eBay vs. Bidder's Edge, Ticketmaster vs. Tickets.com and mySimon vs. Priceman.

6.1 Current Legal Mechanism for Database Protection

Traditionally, the protection of databases generally falls under the realm of copyright act. Patents and trademarks are irrelevant whereas trade secrets are not applicable for databases that are already in the public domain, like publicly accessible web sites. As such, databases are often interpreted as "compilation" in accordance to the copyright law, which defines a compilation as "a work formed by the collection and assembling of preexisting materials or of data…"[33]

To evoke the use of copyright protection, database makers have to make sure their products exhibit a certain degree of "originality" in order to qualify as "original works of authorship". This means certain level of originality or creativity has to be shown in the selection and arrangement of the data. The principle behind this requirement is best illustrated by the landmark case of Feist Publications vs. Rural Telephone Co. in 1991. In that case, phone directory publisher Feist took published its own directory which contained some 8000 records from the phone directory of another company, Rural Telephone Co. The case was decided in favor of Feist because the Supreme Court found that the alphabetical listings of Rural's directory was entirely obvious, and did not constitute a copyright protection based on the "originality" and "creativity" criteria. In a way, it is difficult for databases to be accorded copyright protection given the relatively few ways of compiling and arranging the content of a database.

Another lesser known legal mechanism used by companies for filing database protection is the "trespass to chattel" rule. Essentially, trespass to chattel allows a property owner (including intellectual property) to ward off unwanted outside interference with their property. However, in order to use this law the owner of the property has to provide evidence that suggests the intruder

is engaged in intentional intermeddling with the chattel, and that there are some cognizable harm done to the chattel, be it physical condition, quality or value[34]. In the context of web wrapping, the web sites in which the wrappers extract information from can be considered as the chattel whereas the wrappers are seen as the intruders.

The application of the trespass to chattel law to the web wrapping arena is however, ambiguous. Several recent cases that used the trespass to chattel principle were rather contradictory. For example, in the eBay vs. Bidder's Edge case, eBay used the principle to argue that the wrappers sent by Bidder's Edge was an unwelcome intrusion to its web server and that eBay had to spend additional resources to guard against such an intrusion to its chattel, thereby inflicting financial harm to itself. EBay further contended that while the Bidder's Edge's wrappers had not yet caused significant harm to its web servers in terms of tying up the server's resources, eBay might attract a lot more other unlicensed wrappers in the future that could severely compromise its severs' performance if Bidder's Edge went unpunished. The court eventually granted a preliminary injunction to stop Bidder's Edge from sending wrappers to eBay's server. Although Bidder's Edge appealed against the sentence, the two parties soon settled out of court, leaving this ruling intact. The statue of the trespass to chattel law as applied to the world-wide-web is further complicated by another development in the Ticketmaster vs. Tickets.com case. The court rejected Ticketmaster's claims that Tickets.com intruded its web site because "it is hard to see how entering a publicly available web site could be called a trespass, since all are invited to enter."[35]

6.2 The European Database Directive

It is against this backdrop of flimsy legal protection for databases in the Internet era that the European Database Directive was enacted in 1996. Based on a sui generis doctrine (Latin for "of its own kind"), the EU Database Directive defines a database as "a collection of independent works, data or materials arranged in a systematic or methodical way and individually accessible by electronic or other means."[36] And since a database can contain any type of information including text, images and music clips, these data formats in both electronic and print versions, are also covered under the directive.

Under the EU Database Directive, a database maker who has expended substantial investment in the compilations can prevent unauthorized "extraction" and "utilization" of all or a "substantial part" of the database by a third party, for a period of 15 years measured from the completion of the database. To prevent users from circumventing the "substantiality" requirement, the directive states that "repeated and systematic extractions of insubstantial parts" are not allowed. However, to enjoy this sui generis protection, a database maker must establish that data extraction or re-utilization by a third party "conflicts with the normal exploitation of the database," undermines the "normal or potential market" or "unreasonably prejudice the legitimate rights of the rights holder."[36]

Perhaps the most controversial aspect of the EU Database Directive lies in its requirement that the database makers must be nationals of the European Union, or have established habitual

residence in the EU, in order to qualify for its protection. This means that while a European database maker can sue anybody for unauthorized data extraction within the EU, his American counterpart will enjoy no such protection if his database is similarly compromised in the EU. The only way for this American database maker to enjoy this sui generis right is for the US to pass similar database protection legislation. This is because according to the EU directive, the sui generis rights can be extended to other countries with similar database protection framework.

The EU Database Directive also contains several clauses that further define the scope of protection extended to original makers of databases. For example, it explicitly excludes subcontractors from the definition of "database maker". This means EU companies are able to stake their claims as the "database maker" even if they outsource the making of databases to a third country such as India or China. However, the Directive does not specify the rights of the employees with regards to the databases that they create. Instead, it defers to the national laws of the member states in governing the "work-made-for-hire" aspect of the database creation.

6.3 HR3261 Database and Collection of Information Misappropriation Act

As a direct response to the EU Database Directive, several congressmen in the US had lobbied for similar database protection bills to be passed, culminating to the current HR3261 bill introduced in October 2003. While the bill is still being debated at the time of writing, the preliminary draft sheds some light on the future of the direction of database protection environment in the US.

The HR3261 Database and Collection of Information Misappropriation Act defines a database as "a collection of a large number of discrete items of information produced for the purpose of bringing such discrete items of information together in one place or through one source so that persons may access them."[37] Under this definition, several items are excluded from this Act including:

1. An original work of authorship that is neither a compilation nor a collective work.
2. A collection of information that are used primarily for digital communication such as routing and forwarding.
3. Audio and video programming.
4. Directory of domain name registrants, unless it is properly maintained to ensure integrity, accuracy and is fully accessible by the public.

Section 3 of the HR3261 contains the clauses of prohibition against misappropriation of databases. It prohibits anyone from making available in commerce a "quantitative substantial part" of the database without the database maker's authorization, if all of the conditions below are satisfied:

1. the database was generated, gathered, or maintained through a substantial expenditure of financial resources or time;
2. the unauthorized making available in commerce occurs in a time-sensitive manner;

3. the making available inflicts injury on the original database or its related service offerings by serving as the functional equivalent in the same market as the database in a manner that causes the displacement or disruption of revenues; and
4. the ability of others to "free ride" on the efforts of the original database maker would reduce the incentive to produce the product or threaten its existence.

At the same time, section 4 of this Act contains a list of permitted acts that include:

1. independently generated or gathered information;
2. acts of making available in commerce by nonprofit educational, scientific or research institutions, provided these acts are reasonable under the circumstances, taking into account the customary practices associated with such uses of such database by nonprofit educational, scientific, or research institutions;
3. hyperlinking of online locations;
4. news reporting, unless the information is time-sensitive and has been gathered by a news reporting entity, and making available in commerce the information serves as direct competition,

Like the EU Database Directive, HR3261 is based on the sui generis doctrine. Sections 3 and 4 combine to give a rather large scope of fair-use exemptions, the major sticking point surrounding past debates of similar bills. By establishing a high threshold (section 3) for which a database maker needs to satisfy before taking legal actions against the defendants, the Act prevents database makers from embarking of costly litigation proceedings against nonprofit educational bodies and research institutions, probably by far the largest users of databases.

In the following sections, we will examine the similarities and differences between HR3261 and the EU Database Directive.

6.4 Comparative Analysis of the EU Database Directive and HR3261

As pioneers of the sui generis database protection doctrine, both the EU Database Directive and the HR3261 (if passed into law) have significant influence on the future direction of the database-producing industry. Their respective success and failure will determine whether or not more innovative and valuable databases will be created, or if the entire industry will be saddled with 'anti-commons' problems where the many rights of exclusion held by different individuals mean no useful databases can be created without incurring substantial loyalty and license fees. To better understand the effects of these two Acts, we need to examine the implications of each of them, and compare the similarities and differences between them.

6.4.1 Definition for Database

The EU Database Directive defines a database as "a collection of independent works, data or other materials arranged in a systematic or methodical way and individually accessible by electronic or other means."[36] On the other hand, HR 3261 states that a 'database' means "a collection of a large number of discrete items of information produced for the purpose of

bringing such discrete items of information together in one place or through one source so that persons may access them."[37]

There is little variation between the definitions of database under each Act. Both Acts ascertain the conventional understanding that a database is made up of discrete elements that can be accessed individually. Both Acts also define database very broadly, and as Lipton would argue, they followed a copyright structure in which the creator is given broad rights with some limited "fair use" exceptions carved out of those rights. The only trivial point that sets the two Acts apart in database definition is that HR3261 states explicitly that a database is a collection of items of information. However, while the EU Directive makes no mention of such a point, it is not expected that charges against non-information items will be filed under this sui generis law in either continent.

6.4.2 Scope of Protection

The EU Directive and HR3261 differ in approaches with regard to the scope of protection accorded to database makers. The EU Directive takes a more pro-active approach by drafting a listing of restricted that prohibits unauthorized legal entity from the following:

a. temporary or permanent reproduction by any means and in any form, in whole or in part
b. translation, adaptation, arrangement and any other alteration
c. any form of distribution to the public of the database or of the copies thereof
d. any communication, display or performance to the public
e. any combination of the above

On the other hand, HR3261 puts the responsibility of database protection squarely on the database makers or owners. As discussed previously, a database maker has to be able to demonstrate that unauthorized extractions of his compilations result in several conditions detrimental to his business. Instead of dictating what kinds of actions should or should not be carried out, HR3261 chooses to focus on the seriousness of consequences that befall on the original database makers in situation whereby should unauthorized extractions occur. This introduces a certain degree of flexibility allowing this Act to remain relevant in the face of future technological changes associated with database creation or extraction. On the flip side, the lack of clearly stated restricted acts means ambiguity is likely to creep in, potentially subject HR3261 to different interpretations by different parties or courts.

HR3261 appears to be a step ahead of the EU Directive in terms of the scope of database protection. Its emphasis on the "harm inflicted" on the database makers displaces the need for the Act to be updated with advancing technological changes. However, at this nascent stage of sui generis database protection, an explicit listing of restricted acts will help clear up the confusion. For example, selling a translated version of an original database in the same geographical market is likely to invite much controversy including the definition of market, and if translated copies that are sold to a different group cause a displacement of revenues. This is especially of concern to the US where the demography of the population is more diverse and different ethnic communities speaking different languages often live in the same physical environment.

6.4.3 Exceptions to Protection

As with traditional copyright laws, the sui generis EU Directive and HR3261 also contain a list of permissible acts under which extractions become lawful. The EU Directive does not explicitly contain a list of permitted acts per se, but provide a set of guidelines for member states to draft their own exceptions. These guidelines suggest that the use of a protected database be allowed for:

a. the reproduction of a non-electronic database for private use
b. illustration of teaching or scientific research unrelated to commercial purposes
c. the purpose of public security for the purpose of an administrative or judicial procedure
d. other circumstances which are authorized under traditional national IP laws

Similar to the EU Directive, HR3261 contains a list of permitted acts that measure the intent of the users. For example, it allows for the generation of an identical database provided that the content of the second database is independently gathered or generated, without reference to the original database. It allows for hyperlinking and non-time-sensitive news reporting. HR3261 also contains a clause that differs somewhat from the EU Directive. Instead of limiting the use of original databases for non-commercial teaching or scientific research, it allows for non-profit educational, scientific or research institutions to profit from the use of such databases.

Both the EU Directive and HR3261 clearly spelled out what could or could not be done with respect to permissible acts. This follows the traditional copyright approach in which a broad categorization of rights is granted to the originator, only for these rights to be limited by detail-specific exceptions. The major difference that stems from this category is the clause in HR3261 that allows for non-profit educational and research institutions to utilize original databases for commercial purposes. However, the HR3261 Act veers into ambiguity yet again when it mentions these exceptions will be determined by courts for which "the making available in commerce of the information in the database is reasonable under the circumstances, taking into consideration the customary practices associated with such uses of such database by non-profit educational, scientific or research institutions and other factors that the courts determines relevant."

6.4.4 Term of Protection

Databases protected by the EU Directive enjoy a 15 years protection. Additionally, any substantial changes subsequent to the creation of the original database automatically renews the term of protection to another 15 years. The kinds of permissible changes include additions, deletion, edition or an accumulation contextual alteration. This means that dynamic databases that are updated periodically will enjoy infinite protection while static databases will be granted a protection of 15 years only. However, it is expected that database makers will attempt to substantially modify their databases at least once during the 15 years period, rendering most databases created under the EU Directive subjects of infinite protection. Only the databases that have no true commercial values will be left to expire.

HR3261 is totally silent about the term of protection, inviting a number of speculations. One of them is that this Act can be interpreted as affording the database makers infinite protection, whether the database a dynamic or static. If this is indeed the case, HR3261 is actually a step backward with respect to the EU Directive. This is because by granting infinite protection to database makers, this Act takes away the incentive for the database makers to properly maintain their products. It is also unprecedented in the in the intellectual property jurisdiction domain where every form of protection is granted a "life expectancy" so that these products help spur further innovation and brings more societal welfare.

6.4.5 Relations to other Laws

The EU Directive is straightforward in saying that the sui generis database protection law does not affect the other traditional laws including copyright, patents, trade secrets, trade marks, data protection and privacy, confidentiality, access to public documents, laws on restrictive practices and the law of contract, amongst others.

HR3261 is unambiguous in its recognition of other rights. It contains a near-identical clause recognizing that the rights of other laws such as copyright patents are not affected.
It also recognizes the Communications Act of 1934 and the Securities Act of 1933. However, it overrides state laws including State statue, rule, regulation or common law doctrine.

The following table is a summary of the differences between the EU Directive and HR3261 as discussed above.

	Parameters	EU Directive	HR3261
1	Definition of Database	a collection of independent works, data or other materials arranged in a systematic or methodical way and individually accessible by electronic or other means.	a collection of a large number of discrete items of information produced for the purpose of bringing such discrete items of information together in one place or through one source so that persons may access them
2	Scope of Protection	i) temporary or permanent reproduction by any means and in any form, in whole or in part ii) translation, adaptation, arrangement and any other alteration iii) any form of distribution to the public of the database or of the copies thereof iv) any communication, display or performance to the public v) any combination of the above	Owners/makers have to demonstrate: i) the database was generated, gathered, or maintained through a substantial expenditure of financial resources or time; ii) the unauthorized making available in commerce occurs in a time-sensitive manner; iii)the making available inflicts injury on the original database or its related service offerings by serving as the functional equivalent in the same market as the database in a manner that causes the displacement or disruption of revenues; and iv) the ability of others to "free ride" on the efforts of the original database maker would reduce the incentive to produce the product or threaten its existence

3	Exceptions to Protection	i) the reproduction of a non-electronic database for private use illustration of teaching or scientific ii) research unrelated to commercial purposes iii) the purpose of public security for the purpose of an administrative or judicial procedure iv) other circumstances which are authorized under traditional national IP laws	i) Independently generated or gathered information ii) Acts of making available in commerce by nonprofit educational, scientific or research institutions iii) Hyperlinking iv) News reporting (non-time-sensitive)
4	Term of Protection	15 years	Not Specified
5	Relations to other laws	Other rights not affected	i) Other rights not affected ii) State laws over-ridden

Table 5: Comparative Analysis of EU Directive and HR3261

5.5 Implications of HR3261

The HR3261 and the EU Database Directive each takes a different approach in defining the level of protection afforded to the makers of databases. For example, while the EU Directive explicitly states what specific arts are to be restricted, HR3261 is more concerned about the seriousness of the consequences brought about by an act of extraction, never mind in what way. For example, the EU Directives prohibits unauthorized users from temporary or permanent reproduction by any means and in any form of the original database, HR3261 only goes as far as implying such arts will be allowed so long as the unauthorized users do not cause harm to the original database makers in terms of displacement of revenue, sales, licenses, advertising, etc. Alternatively, extraction of this nature will also be allowed if the database is not generated or maintained through a substantial expenditure of financial resources or time.

Although the approaches taken by each of these Acts differ, their collective implications on the database industry are beyond doubt. Despite the good intention of this act to promote an accelerated growth of database creation, many academia and lawyers remain critical on its long-term effects on the whole industry. In what follows, we will present a list of common critiques[34,35] on the implications of these database protection measures, with particular emphasis on the HR3261.

1. **Duplication of work**

By making the content of a database copyrightable, a second comer would be fearful of breaking the laws by making use of preexisting materials. He would have to go back to the original source to extract the data that he needs. Thus, instead of expending his effort and resources into creating more value-added features or services, a considerable amount of his investment might go into reproducing the same piece of information, especially when the licensing cost is prohibitive. This runs counter-intuitive to most traditional norms where the use of preexisting data in generating new products or services is strongly favored. This is especially true in the world of academia.

2. **Monopoly of data by the first compiler**

It is also possible that the first compiler might abuse this right granted upon him. Drawing on the Feist example, Rural staunchly refused to license the content of its database to Feist because by doing so, Rural is able to maintain its lucrative yellow pages advertisement revenue. By forcing Feist to go back to the original source which requires very substantial effort and resources (in this case, Feist would have to spend much more than Rural because of Rural's unique position as the service provider), Rural was able to effectively monopolize the local telephone directory market. This would result in a form of market failure because a product would have to be produced at a considerable higher cost than necessary. The likely result of that would be a higher advertising cost if Feist's directory went to the market, or a higher advertising costs for Rural's yellow pages since it would be able to extract rent through its monopolistic position. The repercussion effect, one would imagine, is an overall welfare reduction to society as higher advertising costs translates to higher retail prices – a suboptimal equilibrium.

3. **Loss of creativity in database design**

Under copyright laws, a database needs to exhibit certain qualities of "creativity" or "originality" in order for it to receive protection. Under this sui generis doctrine, almost any database will be eligible for legal protection so long as a "substantial" amount of efforts or financial resources have been expended in generating the database. The likely result of this is that manufacturer will dispense away with adding more "intellectual sparks" in creating databases. This means the databases so created will tend to be of lower quality because of the lack of interest or incentive to make them either easily accessible or other associated benefits that come with a more creative arrangement of databases. This point is especially detrimental for databases that reside on the web (and for which web wrappers prey on) because a poorly arranged database means users (or software agents) will find it harder to navigate around to extract useful information, not to mention the hidden costs in assessing a plethora of poorly designed databases, such as additional time and effort wasted.

There are many more arguments pertaining to the negative impacts that a sui generis protection on database will bring about, but the most compelling ones centered on the potential loss of intellectual creativity and a possible stiffening of the knowledge as preexisting or factual materials become increasingly harder to access. On the other hand, several optimistic views are being raised about the positive effects that such legislation may bring about, including:

1. **More knowledge creation through more investment on database**

Because the sui generis doctrine is based on economic ground, legal protection afforded to the content of databases is likely to bring about an increased level of investment into data compilation. This is likely to stimulate knowledge creation as more data or information becomes more widely available. However, an increase level of knowledge creation is only likely when the costs of acquiring the data/information are not prohibitive. Given that the HR 3261 Act states that "substantial" investment has to be made before a certain database is afforded protection, the licensing cost is generally thought to be rather significant. If this is indeed the case, the

prohibitive costs will actually deter people from using the database and precious resources might be "wasted" as independent research on data collection is carried out to avoid the hefty licensing fee. The result of that is an overall decrease in investment in new knowledge generation as the legal forces act counter to the free exchange or sharing of available data/information.

2. Better protection for American databases with respect to EU Directive

HR 3261 is a direct response to the EU Directive on Database introduced in 1996. One of the clauses in the EU directive that worries many American lawmakers is that foreign database is not protected at the same level as the European databases in Europe unless the host nations adopt a similar level of domestic protection. This effectively means that unless the U.S. (and other countries) adopts a similar measure as the EU directive, databases maintained, owned or operated by U.S. citizens will not be receive this sui generis protection in Europe. At the same time, anyone (including U.S. citizens) who are found to have "inflicted an injury" on a European database will be subjected to the more stringent database laws. As such, the EU directive essentially creates an adverse selection phenomenon where people countries will vie to introduce yet tougher laws on database protection in order to best serve the interest of their citizens. Interestingly, HR3261 is not perceived as more stringent than the EU directive. However, it probably has to do with the strong opposition from the scientific and research communities here lobbying against strong measures to curb free use of database rather than as a direct consequence of market forces.

Whether it is to increase knowledge creation by awarding database compilers or to better protect the owner of American databases, HR 3261 faces many challenges if pass into laws. Recent statistics suggest that the European experience with database protection is not at all smooth sailing as the number of databases created surges initially before plummeting as the protective measures hinder further progress. No substantial database can be created without having to rely on preexisting data sitting somewhere in other databases.

5.5 Recommendations

The HR3261 is a step in the right direction in achieving the delicate balance between over-protection of database makers and under-utilization of the same databases which may inadvertently result in anti-commons problem. In the following section, we provide a list of recommendation that supplements the existing database protection framework, with particular references to web aggregation services, the main subject of contention for database misappropriation claims and a heavy user of web wrapping technologies.

Recommendation 1: Engaging "authorized" Aggregators

While most web sites see web spiders as unwanted intrusion, there are many instances where web mining by an aggregator can actually bring more good than harm. For example, a consumer electronic store (such as PC Mall) selling the Apple iPod probably wants an aggregator such as mySimon.com to actively mine its database so that it can reach a larger pool of customers

through these "sales" channels. In this case, formal agreements can be struck between PC Mall and mySimon.com so that it is able to obtain accurately updated price information directly from PC Mall, perhaps through a different servers independent from the one where the store customers use. This creates some sort of an electronic data interchange, or EDI where only authorized web aggregators have access to. If PC Mall is able to devise a strategy to price discriminate such that mySimon.com users may enjoy a higher discounted price compared to other non-authorized aggregator, such an arrangement will be beneficial to both parties. This is because mySimon.com will be able to help PC Mall expands its reach to potential customers while preferential discounts from PC Mall will make mySimon.com more attractive than its competitors. By directing mySimon's and other "authorized" aggregators' traffic to a different server, PC Mall will be able to ensure the service level of its main servers where its customers use.

Recommendation 2: Re-emphasize more on common laws such as contract law

Apart from copyrights, contract laws are also used to resolve disputes arising from data appropriation in cyberspace. By tying a user to a "click-wrap" license agreement, an aggregatee will be able to exercise greater control in determining who will be allowed to access the information and in what ways. However, for such license to be binding, the web site must be able to demonstrate that the user is aware of the license and that the user must be able to accept it in some way, as illustrated in the precedent-setting Specht et al. v. Netscape Communications case. This kind of arrangement is particularly relevant to account aggregation should the aggregatee chooses not to let its customers to share their account information.

The flip side of this arrangement is that sometimes these aggregatees do not want to upset their customers. While the terms and conditions set forth in the license agreement might be legally binding, most aggregatees chose not to pursue legal actions against their customers for fear of upsetting them. For example, most airlines have some sort of clauses in their license agreements that prohibit members of their frequent flyer programs from disclosing account information to third parties. However, not many of them diligently enforce these clauses when their customers choose to disclose their account information to third-party aggregators such as Maxmiles, for fear that these upset customers might decide not to fly with them anymore. Despite its limitations in this regards, contract laws are still very effective in ensuring the content of the databases are not exploited commercially by "rogue" users such as Michael Zeidenberg.

Recommendation 3: Use of subscription based premium content

Where the database or compilation is the result of substantial capital investment, database manufacturers have the tendency to limit access in order to extract "rent" from users accessing the databases that they created. One of the possible solution is to follow a common industry practice by collating these "premium" data into a secured site and limit its access through subscription, much like what the Wall Street Journal is doing with its online content, i.e. users only get to read WSJ's articles when they are given a password after subscribing. Access to these premium database or information can be further protected through a "click-wrap" license agreement, binding users from making duplicate copies.

The use of subscription based premium content sends out two signals: one is that it signifies the value of these databases therein contained; the other being it is an explicit declaration that the content is not for free public access and therefore not "all are invited to see". This arrangement is likely to separate genuine and serious compilations works from the likes of online catalogs since market economics will dictate the true values of the content of these databases, i.e. nobody is likely to pay to view an online catalog whereas the listed prices of a range of blue chip stocks over the past 20 years are likely to generate substantial interest and buying activities.

6.6 Conclusion and the Future of the Database Protection

The main drawback of HR3261 is its ambiguity. We feel that HR3261 leaves too much unanswered questions that are of great concerned to a lot of stakeholders in this database producing industry. For instance, it does not address the term of protection. In contrast to the EU Database Directive, HR3261 also looks light in terms of details. As a new bill that built on a sui generis doctrine, we expect the bill to be more straightforward and clear-cut, with specific details that address the concerns of the stakeholders. The resistance and critique for this bill is unlikely to go away so long as key questions remain unanswered and some parts of the bill appear equivocal. It is therefore important for the bill to state what is to be done and what is not to be done, especially at early stages. In that regard, a model that is similar to the EU Directive where much details are given is probably more preferable in this nascent stage.

However, by emphasizing on the intent of the "extractors" of the databases rather than the actual physical actions taken by extractors, HR3261 has taken a step in the right direction. The fact that HR3261 does not impose a residency requirement and therefore raises the bar even further is a welcome relief. We expect that in the future a new sui generis right for database protection will be enacted throughout the world, much like the other traditional intellectual property protection measures like copyrights and patents. Time will allow this sui generis doctrine to find an equilibrium point in the law books that is both consistent with the current legal framework and acceptable by the educational and research community at large.

Chapter 7: Conclusion

In this thesis, a total of 20 commercial and academic web wrapping toolkits were studied, their architecture described, and their features analyzed and compared, using a two-tier taxonomy. Legal/policy developments related to web wrapping technologies were also discussed, with particular emphasis on the EU Database Directive by the European parliament and the HR3261 bill by the United States Congress.

The 1st-tier taxonomy used a "bottom-up" approach by breaking down each of the 20 toolkits under study into its respective functional features, such as the degree of automation of web wrapper creation and the support for SQL. Of the 20 toolkits, an overwhelming 14 of them made use of some sort of semi-automatic approach for wrappers creation, indicating the use of such technologies is gradually transferred from skilled programmers to the general public. The presence of powerful web wrapping toolkits such as Lixto's Visual Wrapper and Kapow's Robosuite further confirms the trend towards automation.

On the other hand, manual web wrapping toolkits remain powerful and provide the greatest degree of flexibility to allow developers to customize the wrappers to extract from complex web pages or documents. Cameleon and WebQL are the representative toolkits in this category. It was also perhaps worth noting that other semi-automatic web wrapping toolkits sometimes also offered the same degree of flexibility, but such flexibility could only be achieved by manual coding in the editor. The sole automatic toolkit in this study, WIEN, was an academic toolkit and at the moment, supported limited features due to its heavy reliance on the developing induction algorithm.

The 2nd-tier taxonomy shed further light on the state of the art of the current web wrapping technologies. With 8 out of 14 semi-automatic toolkits supporting all of the 12 selected features, it was testimonial that web wrapping technologies had come of age to be adopted by mass consumers. At the same time, 2 out of 5 manual web wrapping toolkits supported only 6 core features. While the price differential might play a role here, this phenomenon nevertheless suggested future manual toolkits might be more specialized in certain niche areas, such as extracting from free text.

The last part of the thesis was a comparative analysis between the EU Database Directive and the HR3261, both of which dealt with the intellectual protection mechanism used to protect "substantially invested" databases that were the subjects of unauthorized extraction. From the analysis, it was found that HR3261 was still fraught with ambiguity, with many details and questions left unanswered. Nevertheless, it represented a new break-through compared to its predecessors by focusing on the extent of injury inflicted on the database owners by unauthorized extractors. HR3261 also deserved credit in that by electing not to impose a residency requirement, it would not be drawn into a vicious cycle of raising the bar for database protection, as intended by the EU Directive. With further modification to its list of permitted and prohibited acts, this bill represented a model that was workable and potentially acceptable by the research and educational community at large.

Reference:

1. Stefan Kuhlins and Ross Tredwell, " Toolkits for Generating Wrappers – A survey of Software Toolkits for Automated Data Extraction from Web sites", Net.ObjectsDays 2002
2. Alberto H.F. Laender et al, "A Brief Survey of Web Data Extraction Tools", ACM SIGMO Record, 2002, p.84-93
3. Aykut Firat, "Information Integration Using Contextual Knowledge and Ontology", MIT Sloan PhD thesis, 2003
4. W3C, http://www.w3.org/DOM/
5. Compaq Corp., http://www.research.compaq.com/SRC/WebL/index.html
6. QL2 Software, http://www.ql2.com/index.php
7. German National Research Center for Information Technology, http://www.ipsi.fraunhofer.de/oasys/projects/jedi/
8. Stanford University, http://www-db.stanford.edu/tsimmis/tsimmis.html
9. Georgia Institute of Technology, http://www.cc.gatech.edu/projects/disl/XWRAPElite/
10. Lixto Software, http://www.lixto.com/
11. Califf, M.E. and Mooney, R.J., "Relational Learning of Pattern Match Rules for Information Extraction", In Proceedings of the Sixteenth National Conference on Artificial Intelligence and Eleventh Conference on Innovative Applications of Artificial Intelligence (Orlando, FL, 1999), p.328-334
12. Soderland, A., and Azavant, F. "Learning information Extraction Rules for semi-structured and free text", Machine learning 34, 1-3 (1999), p.233-272
13. University College Dublin, http://www.cs.ucd.ie/staff/nick/home/research/wrappers/wien/
14. I. Muslea, S. Minton and C. Knoblock, "Hierarchical wrapper induction for semistructured information sources", Autonomous Agents and Multi-agents Systems 4, ½ (2001), p.93-114
15. B. Adelberg, "A Tool for Semi-Automatically Extracting Structured and Semi-Structured data from text documents", in Proceedings of the ACM SIGMOD International Conference on Management of Data (Seattle, WA, 1998), p.283-294
16. Kapow Technologies, http://www.kapowtech.com/
17. Brigham Young University, http://www.deg.byu.edu/
18. Fetch Technologies, http://www.fetch.com/
19. ItemField, http://www.itemfield.com/
20. IBM, http://www-306.ibm.com/software/data/eip/features_infomining.html
21. WebMethods, http://www.webmethods.com/
22. ShueTech, http://shuetech.com/minetheweb/
23. NQL Technologies Inc., http://www.nqltech.com/2004/index.asp
24. Kapow Technologies, "Robosuite Technical White paper", http://www.kapowtech.com/
25. ExtraData Technologies, http://www.extradata.com/
26. Altercept, http://www.altercept.com/
27. Lencome Software, http://www.lencom.com/
28. Cannotate Technologies, http://www.connotate.com/default.asp
29. Loton Tech, http://www.lotontech.com/
30. Thunderstone, http://www.thunderstone.com/texis/site/pages/Home.html

31. WinTask, http://www.wintask.com/
32. XSB Inc., http://www.xsb.com/index.html
33. Copyright Office, http://www.copyright.gov/register/tx-compilations.html
34. P. Samuelson, “Legal Protection for Database Contents”, Communication of the ACM, December 1996, Vol 39, No.12
35. H. Zhu, S. Madnick and M. Siegel, “The Interplay of Web Aggregation and Regulation”, 2003
36. THE EUROPEAN PARLIAMENT AND THE COUNCIL OF THE EUROPEAN UNION, http://europa.eu.int/ISPO/infosoc/legreg/docs/969ec.html
37. 108th Congress, http://www.haledorr.com/files/upload/data_collections_act_2003.pdf