

Towards Building Logical Views of Websites

Zehua Liu^a Wee Keong Ng^{a,*} Ee-Peng Lim^a Feifei Li^{b,1}

^a*Centre for Advanced Information Systems, School of Computer Engineering,
Nanyang Technological University, Singapore 639798, SINGAPORE*

^b*Computer Science Department, Boston University, Boston MA 02215, USA*

Abstract

Information presented in a Website is usually organized into certain logical structure that is intuitive to users. It would be useful to model websites with such logical structure so that extraction of Web data from these sites can be performed in a simple and efficient manner. However, the recognition and reconstruction of such logical structure by software agent is not straightforward due to the complex hyper-link structure among webpages and the HTML formatting within each webpage. In this paper, we propose the WICCAP Data Model, an XML data model that maps websites from their physical structure into commonly perceived logical views. To enable easy and rapid creation of such data models, we have implemented a visual tool, called the Mapping Wizard, to facilitate and automate the process of producing WICCAP Data Models. Using the tool, the time required to construct a logical data model for a given Website is significantly reduced.

Key words: Web Information Extraction, Web Data Model, Supervised Wrapper Generation

PACS:

1 Introduction

The World Wide Web has become such a successful channel in delivering and sharing information that people are getting used to searching the Web as the first resort for information. However, users are often not able to retrieve

* Corresponding author

Email addresses: liuzh@mail.ntu.edu.sg (Zehua Liu), awkng@ntu.edu.sg (Wee Keong Ng), aseplim@ntu.edu.sg (Ee-Peng Lim), lifefei@cs.bu.edu (Feifei Li).

¹ Work done while in Nanyang Technological University, Singapore

exact information of interest. The amount of data accessible via the Web is huge and growing rapidly. *Information overloading* has been identified as the problem, as opposed to the previous problem of information unavailability. It is usually a time consuming and tedious process for users to finally get the right information that they are looking for.

To address this problem, over the past couple of years, some *Information Extraction* (IE) systems, mostly in the form of wrappers and agents, have been developed to automatically extract target information. In the context of the World Wide Web, the key issues of information extraction include defining a flexible data model for representing information from the Web, specifying where the desired information is and how to extract the information, extracting and processing the information, and presenting the extracted information in an accessible manner for further processing or integration.

As the first step, defining data models for mapping between the actual information on the webpages and the pieces that the users are interested is important for an IE system. This process usually requires in-depth technical knowledge and, if done manually, is difficult and time-consuming. In addition, the rate at which new websites are appearing and old websites are changing is much faster than what the current IE systems can handle, even if expert users are the target users. Therefore, helping users quickly create the required data model for extracting information has become the top priority of IE system for the Web.

Information presented in a website is usually organized according to certain logical structure that corresponds to users' common perception. If the data models that IE systems produced reflect this commonly perceived structure, most users will be able to understand and use them easily. Unfortunately, the data models of most current IE systems [1,5,8,15,18,23,24] either only deal with information in page-level or represent the extracted information in some proprietary and ad-hoc data structures that do not truly reflect the original logical view of the websites.

1.1 The WICCAP Approach

In this paper, we propose a data model, called the WICCAP Data Model (WDM), to map information from a website into its commonly perceived organization of logical concepts. WDM is designed to model information from the Web at site-level so that the data extracted according to this model represents a complete view of the original site and no additional efforts are required to further integrate data from different pages. The aim of WDM is to enable ordinary users to easily understand the logical structures of a site in order to

perform extraction tasks without worrying about the technical details. WDM is proposed as part of the *WWW Information Collection, Collaging and Programming* (WICCAP) system [17,19,20], which is a Web-based information extraction system to enable people to obtain information of interest in a simple and efficient manner.

The approach taken by WICCAP is to explicitly separate the tasks of information modeling and information extraction so as to allow ordinary users to perform information extraction. In WICCAP, expert users construct the WICCAP Data Model and specify how to extract the information; ordinary users indicate what to extract based on the given WDM and use the tools provided by the system to extract and view the information.

Besides being easy-to-understand and easy-to-use to ordinary users, WDM is designed to be *flexible* enough to work with heterogeneous categories of websites and *open* enough to allow other systems and applications to understand and make use of it.

To cope with the rapid changes of old websites and establishment of new websites, a software tool, called the Mapping Wizard, has been implemented to assist users to quickly generate the data model of a given website. The focus of the Mapping Wizard is on how to quickly generate a logical data model for a given website. To do this, we formalize the process of creating a WICCAP Data Model for a given website to permit as much automation as possible and develop assistant tools and algorithms to help automate the process of creating data models.

1.2 Overview of Paper

The remainder of the paper is organized as follows. Section 2 elaborates the logical data model in detail. Section 3 then presents the process for creating such data model. Section 4 describes how the tool helps users to quickly generate data models. Some experimental results are presented and discussed in Section 5. Section 6 compares our work with related systems. Finally, Section 7 gives some concluding remarks.

2 WICCAP Data Model

In most current Web Information Extraction systems, when defining data models, users usually have to directly specify which portion of a webpage within a website constitutes the target information. The isolated pieces of

target information are later re-organized to make them more readable and accessible. As a result, the data model produced is usually not intuitive to other users and is very specific to the particular website.

In WICCAP, we try to derive a logical data model (WDM) of the target website first and then extract information from the website based on this model. The role of the WICCAP Data Model is to relate information from a website in terms of a *commonly perceived logical structure*, instead of physical directory locations. The logical structure here refers to people’s perception on the organization of contents of a specific type of website.

For example, when a newspaper website, such as BBC Online News [6], is mentioned, people generally think of a site that consists of sections such as World News, or Sports News; each section may have subsections and/or a list of articles, each of which may have a number of attributes such as title, summary, the article itself, and maybe links to other related articles. This hierarchy of information is the commonly perceived structure of a newspaper website, which most users are quite familiar with. If we model and organize the information from a newspaper website in this way, the resulting data model will appear familiar to and be easily accepted by most users. In addition, since this logical structure is applicable to the entire category of websites, e.g. all newspaper websites, it can be re-used in the future when creating data models for websites of the same type.

The logical model is essentially a tree, where each node represents a logical concept perceived by the user. The organization of individual tree nodes into a tree forms the logical skeleton of the target website. This logical structure is what the users of the next layer in WICCAP (for information extraction) will be operating on. In order to allow extraction to occur, each tree node contains an element named “*Mapping*” (discussed in detail in Section 2.3) that associates the node with the corresponding data in the actual website. This *Mapping* element in fact represents the extraction rule(s) that enable the extraction agent to perform the actual extraction of information. For leaf nodes of the tree, the *Mapping* element points to the data item that the user wants to extract. For example, a leaf node “Title” may have a Mapping element that contains the extraction rule to extract the title of the news from the webpage. For internal nodes, the Mapping element refers to the portion of the website that contains the desired data to be extracted by all the leaf nodes of the subtrees rooted from those internal nodes. By assigning a Mapping element to each logical node, the logical view of the target website can be decoupled from the physical directory hierarchy and page layout of the original website, while the mapping between the two is still maintained.

An important feature of WDM is that it is able to model not only information in a single webpage, but also in a set of webpages. This set of webpages



Fig. 1. Logical View of BBC Online News

may or may not have similar layout structures in terms of HTML syntax. It could be a small collection of webpages located in the same directory, a whole website, or pages across several websites, so long as all pages together form a uniform logical view. Typically, WDM will be used to model a website or a self-contained portion of a large website.

Figure 1 shows a portion of the logical view of BBC Online News. In this model, we have a root node, under which there are different sections, such as “World” and “Business”. There may be subsections under each section. Each node (Section or Subsection) may have a list of articles, denoted by the node “ArticleList” with a child node “Article”. Attributes and contents of the article are represented as child nodes of “Article”. The mapping information is stored as an attribute of the elements to hide the technical details from ordinary users and to maintain the logical overview of the data model.

2.1 WDM Schema and Mapping Rules

A WDM representation of a specific website is also called a mapping rule, since it is to map the physical structure to the logical one. Each mapping rule represents a logical tree with one Mapping element for each tree node. Every mapping rule has a schema called WDM schema. A WDM schema defines the possible structure that a mapping rule could have. Mapping rules are instances of their WDM schema. The relationship between a WDM schema and a mapping rule is similar to the Class-Object relationship in object-oriented concept. A WDM schema is defined using XML Schema [25], while mapping rules are defined as XML document. This makes mapping rules well-structured, interoperable, and easy to process.

The WDM schema was first designed to describe the logical structure of a category of websites that exhibit similar logical view [19]. We have extended and generalized the definition of WDM schema to make it suitable for representing the logical view of any website. Thus there is now only one global WDM

schema that defines the syntax that all mapping rules should follow. In the remainder of this paper, this global WDM schema is referred to as *the WDM schema*.

All elements defined by the WDM schema can be classified into two groups: elements for logical modeling purpose and elements for mapping purpose. The former refers to the elements constituting the logical tree, whereas the latter refers to the group of elements defining the mapping between the logical element and the actual Web data to be extracted.

2.2 WDM Elements for Logical Modeling Purpose

There are four elements for logical modeling: *Item*, *Record*, *Region*, and *Section*. These elements serve to represent some logical concepts in a mapping rule. Each element corresponds to a logical node and all elements in a mapping rule together form the logical tree. The actual semantic meaning of each element in different mapping rules depends on the websites. This semantic meaning can be partially reflected by setting the values of attributes such as “Name” or “Desc”. Each element also contains a Mapping element that describes how to link the element to the corresponding portion on the website.

2.2.1 Item

An *Item* represents a piece of information that the user is interested in. It is the atomic unit of all the information in the logical data model. In the logical tree, Items are represented as leaf nodes. An Item contains a Mapping element that indicates where is the data to be extracted, and other attributes that describe its semantic meaning. In a mapping rule, only the data extracted according to Items will be eventually stored as the extraction results. The data extracted from other elements (internal nodes) is only treated as intermediate results to be worked on for processing of subsequent extraction rules.

2.2.2 Record

In most cases, we do not just specify a single Item to extract in an entire logical tree. Within a website, it is likely that there will be a lot of Items that the users would like to extract. Typically, some data will appear to be logically related to each other. Thus, we introduce the *Record* element to group several related Items together.

A Record contains a set of Items, or a tuple of attributes. It is usually used to repeatedly retrieve data in a table or a list when its “Repetitive” attribute is

‘True’. It makes WDM more flexible and expressive by allowing it to describe iterative structure, which appears frequently in complex logical representations. As with other main elements, there is a Mapping element to map the Record to the actual webpages where all the data specified by the Items exist.

An example of Record is the Article element in the BBC News mapping rule, with the different attributes of the article, such as Title, Link, Short Description and Content, forming the group of Items under this Record. This article record is iterative since there can be more than one article under each section.

Although Items within a Record are listed in certain order, the Mapping element of each Item (i.e. the extraction rule) is independent of each other’s. This allows Record to handle some variation of the target webpage, such as missing Items and varying-order of Items.

2.2.3 *Region*

A *Region* is defined to help to identify the area where an iterative Record repeats itself. A Region is typically used together with a “Repetitive” Record to repeatedly extract a list of Records. It can be considered as the rough area containing a list of repeated set of data that we are interested in the website.

2.2.4 *Section*

A Record corresponds to a tuple while a Region represents a list. They are only able to model websites with flat structure (i.e. one level). To make WDM more flexible, a *Section* element is provided to allow nesting of elements, which eventually enables WDM to model a more general form of hierarchical tree-like structure that most existing websites exhibit.

A Section is defined to contain a set of Regions together with a set of nested Sections. Both the Regions and the Sections are optional and can interleave each other. But for the Section element to be meaningful, it should not be empty. One example of using Section to create hierarchical structure is depicted in Figure 1, where the Section “World” contains an “ArticleList” (Region) and two subsections “Asia-Pacific” and “Americas”.

The root node of the logical tree is in fact just a Section element. However, to distinguish it from other Sections, we have decided to give it a different name as *Wiccap*. Since the Wiccap element is just a Section element with a different name, we shall not treat it as a new type of element.

With the introduction of Section, we extend the definition of Record to include Section as possible child elements of a Record. This means that a Record will

have a list of disjunctions of Item or Section. This will allow WDM to handle the situation where a record may contain a list of repeating sub-items, for example, a news article with a list of related news.

2.3 WDM Elements for Mapping Purpose

With all the four logical elements, the logical view of a website can be composed. However, this is just a representation of users' perception of the structure of the website. It does not contain any instruction that allows the extraction of data to occur. To make the extraction possible, we defined a set of elements dedicated for mapping from the logical tree node to the corresponding portion within the actual website. These elements are the basic constructs of the extraction rules used in the WICCAP Data Model.

The WDM elements for mapping purpose include: *Locator*, *Link*, *Form*, and *Mapping*.

2.3.1 Locator

A *Locator* is the fundamental element in WDM. It locates a portion within a webpage. This element corresponds to a single-slot extraction rule (in information extraction terminology) [22]. The current WICCAP system only implements a ByPattern Locator that relies on the starting pattern and ending pattern to locate a specific portion that resides between these two text patterns. The expressive power of such extraction language is obviously restrictive. However, it should be pointed out that the more complex the extraction language is, the more difficult it is for users to manually write and for tools to automatically derive the rules. This is the main reason that only a simple "ByPattern" language is chosen in our data model.

Although the current implementation uses the simplest type of extraction language, the definition of the WICCAP Data Model does not exclude the usage of other more powerful languages. Since Locator is the only mapping element that contains the extraction rules, it is possible to import external extraction languages into the definition of Locator, while keeping the rest of the data model untouched. This means that more expressive languages, such as STALKER [23], HEL [24], Elog [5] and context-based languages [3], could potentially be included, if the corresponding implementation is available. But the derivation of such rules within the framework of the Mapping Wizard tool (discussed in Section 4) remains to be investigated.

With a Locator, we are able to retrieve any part within a given webpage, regardless of whether it is a link, attribute, element name, or text. This is the

```

1:      <xsd:complexType name="LocatorType" content="elementOnly">
2:          <xsd:choice>
3:              <xsd:element name="LocatorPattern" type="LocatorPatternType"/>
4:              ... ..
5:          </xsd:choice>
6:          <xsd:attribute name="Type" use="required">
7:              <xsd:simpleType base="xsd:NMTOKEN">
8:                  <xsd:enumeration value="ByPattern"/>
9:                  ... ..
10:             </xsd:simpleType>
11:          </xsd:attribute>
12:      </xsd:complexType>
13:      <xsd:complexType name="LocatorPatternType" content="elementOnly">
14:          <xsd:sequence>
15:              <xsd:element name="BeginPattern" type="BeginPatternType"/>
16:              <xsd:element name="EndPattern" type="EndPatternType"/>
17:          </xsd:sequence>
18:      </xsd:complexType>
19:      <xsd:complexType name="BeginPatternType" base="xsd:string">
20:          :
21:          :
22:          :

```

Fig. 2. Definition of the Locator Element

most fundamental functionality that any information extraction system should have. The definitions of Locator and some of its relevant children are depicted in Figure 2, where a Locator is defined to contain a *LocatorPattern* (line 2–5), which in turn contains two elements (*BeginPattern* and *EndPattern*) (line 13–18) that define the left and right patterns enclosing the target data item to be extracted (see line 19 for the definition of *BeginPattern*). The *LocatorPattern* element is put inside a *choice* under *Locator* (line 2–5) because in our initial design, a *Locator* may be of other types, such as *ByPath* (i.e. using DOM-tree based extraction patterns). But the current implementation only handles the *ByPattern* type of *Locator*. The same reason applies to the *Type* attribute of *Locator* (line 6–11), where the set of possible attribute values could include other values (e.g. *ByPath*).

2.3.2 Link

The ability to extract information across multiple webpages is critical to WDM, since it models information at site-level. A *Link* element corresponds to the concept of a hyperlink in the Web context. It gives WDM the ability to follow hyperlinks in webpages; thus, traversing the entire website. Isolating the concept of a link out of the extraction language definition is one of the distinguishing features of the WICCAP Data Model, since most Web data extraction systems either can not handle multiple pages or have the “linking” instruction embedded inside their extraction languages.

When a *Link* appears in a mapping rule, it indicates that there is a hyperlink in the actual webpage that should be followed. When the extraction agent follows a *Link* in a mapping rule, it attains the same effect as a user clicking on a hyperlink in a browser and being brought to the new page pointed by

```

1:      <xsd:complexType name="LinkType" content="mixed">
2:          <xsd:choice>
3:              <xsd:element name="Locator" type="LocatorType"/>
4:              <xsd:element name="Form" type="FormType"/>
5:          </xsd:choice>
6:          <xsd:attribute name="Type" use="required">
7:              <xsd:simpleType>
8:                  <xsd:restriction base="xsd:NMTOKEN">
9:                      <xsd:enumeration value="Dynamic"/>
10:                     <xsd:enumeration value="Form"/>
11:                     <xsd:enumeration value="Static"/>
12:                 </xsd:restriction>
13:             </xsd:simpleType>
14:         </xsd:attribute>
15:     </xsd:complexType>

```

Fig. 3. Definition of the Link Element

that hyperlink.

As shown in the definition in Figure 3, a Link can be *Static*, *Dynamic* or a *Form* (defined in line 9–11). A static link is just a simple fixed URL given by users when they create the mapping rules. It is typically used to indicate the base address of a mapping rule. For example, in the root element of the BBC mapping rule, we specify a static Link with a value “http://news.bbc.co.uk/”. Although not listed in the choices in line 3–4, a static link is just a simple text string and is allowed in XML syntax by defining the content attribute to be *mixed* (line 1). A Dynamic type of Link’s link value must be extracted by the extraction agent at run-time. The Link element contains a Locator that locates the hyperlink to be extracted. The agent will follow this link to reach another webpage to continue the extraction. This Dynamic type of Link is critical to a flexible and reliable extraction language because the links in a webpage are likely to change while the positions where these links appear remain relatively stable. A good example of dynamic Link is the links of news headlines that point to the detailed news.

2.3.3 Form

The third type of Link is to be generated from HTML forms. The *Form* element caters to a special group of HTML tags: FORM and other related HTML elements. Form is defined as a type of Link because a FORM ACTION causes the server to perform some operation and return another page; this, to the users, has the same effect as clicking on a hyperlink. Typical examples where the Form element may be useful are the search function and user login authentication.

When the type of a Link is Form, the URL is dynamically constructed by concatenating all the name and value pairs of the input elements contained in the specified form. The input parameters are obtained from the user by dynamically constructing a form that assimilates that of the original HTML

form. The extraction agent then posts this composed URL to the remote Web server and obtains the returned webpage. This ability to handle HTML forms is important for a Web IE System due to the extensive use of forms in websites. It is also one of the characteristics that distinguish WDM from other similar systems, as currently only few of these systems [4,24] take forms into consideration or directly incorporate them into their systems.

2.3.4 Mapping

Locator allows us to access anywhere within a single page while *Link* enables us to jump from one page to another. The *Mapping* element combines these two basic elements to allow us to navigate throughout an entire website, or even the whole World Wide Web.

A Mapping is a simple container element that holds either a Link, a Locator or both. A Mapping may also contain a child Mapping element. The definition of the Mapping element can be seen as a higher level extraction language based on the primitive single-slot language provided by Locator and the hyperlink concept given by Link.

The recursive definition of Mapping allows WDM logical elements to jump through multiple levels of links and essentially makes it possible to completely separate the logical structure from the physical structure defined by the hierarchy of links and physical directories. With the Mapping element, it is not necessary that a child logical element (e.g. Region) must be contained within the area on the same webpage as defined by its parent logical element (e.g. Section). The child element can actually be several clicks of hyperlink away from the parent element.

Combining the definition of all the logical and mapping elements described above, we obtain the complete WDM schema. All the mapping rules conform to this WDM schema. In a mapping rule, every logical element, which is shown in the logical tree view as a node, should have a Mapping element to indicate how to get to this node in the actual website. When each logical element that constitutes a logical structure is augmented with a Mapping element, this logical structure is considered as a complete mapping rule that can be used for extraction of information.

2.4 An Example on BBC Online News

In this section, we illustrate with an example of how the WICCAP Data Model translates the physical structure of a website into a logical view. The website that we will be looking at is the BBC Online News [6]. The mapping rule

for this website is shown in Appendix A. Due to the limitation of space, only details of one Section and one subsection of the mapping rule are shown. The Section element here represents a newspaper section such as “World News”. Region and Record correspond to ArticleList and Article respectively. With such semantic meaning associated with each logical element, the schema actually represents the logical view of most of the online newspaper websites.

In the root element, the XML Schema file is specified; this is also the file that contains the definition of the WDM schema. The Mapping of the root element Wiccap is a static Link that points to “http://news.bbc.co.uk”. This is the case for most mapping rules because a website needs to have a base URL or home URL for any navigation to start from.

Under the root node “Wiccap”, there are a few Sections. In this example, we specify “World”, “Business” and so on. Again, the “World” Section has a Mapping, which extracts the link to the page containing the actual “World” section on the website. An ArticleList (Region) is included under the “World” Section to indicate that we are interested in some content in this section. The Article (Record) and Item elements under this ArticleList node give details of the contents to be extracted. In this subsection, for each article (the Record whose attribute Name is valued *Article*), there are three Items specified: the title, the short description of the article and the URL to the full article.

Two Sections “Asia-Pacific” and “Americas” are inserted after the ArticleList. The internal definition of these two subsections are similar to the Section “World”. All the Sections and ArticleLists together form a tree structure of the logical view (Figure 1), which is quite close to our common understanding of a newspaper: a hierarchy of sections and subsections with articles in each section.

3 WDM Creation Process

The manual process of creating any data model for Web information sources, including the WICCAP Data Model, is tedious and slow. To cope with the large number of new and changing websites, we formalize the process of creating mapping rules with the help of the Mapping Wizard.

The formal process of mapping rule generation consists of four stages: *Basic Logical Structure Construction*, *Content Extraction Definition*, *Mapping Rule Generation*, and *Testing and Finalization*.

Basic Logical Structure Construction. In this stage, the user focuses on building the overall logical structure instead of the details of how to map the

logical tree nodes into the actual website. To start constructing the logical view of a website, a user supplies the home URL of the target website, such as “http://news.bbc.co.uk” to the Mapping Wizard. It then figures out the basic logical organization either by using the user’s knowledge and understanding of the website or by navigating the website. A logical tree can be constructed with the understanding of the organization of the content provided by the website. In the example in Appendix A, this step would involve browsing the BBC website and creating a tree that contains **World**, **Business**, etc as Sections at the first level and **Asia-Pacific**, **Americas**, etc as Sections under the World Section. The Regions, Records and Items under each Section would be created as well.

The basic logical structure built in this step consists of the WDM elements for logical modeling purpose. As we mentioned earlier, websites of the same category share similar logical structure. Although currently not implemented in the Mapping Wizard, it is possible to re-use (at least part of) the basic logical structures constructed for other websites by retaining the relevant WDM elements for modeling purpose in a mapping rule generated earlier for a similar website. With some modification effort, the user can proceed to the next step to specify the mapping details.

Content Extraction Definition. After the basic logical tree has been clearly defined, the user continues with the *Content Extraction Definition* stage to define specific information that is of interest and relates the nodes in the logical tree to the actual website. For instance, the Mapping elements (shown in Appendix A) for each element created at the step earlier (e.g. the **World Section** element and the **Item** elements) would be discovered and defined at this step.

As described earlier, the Mapping element uses Link and Locator to eventually point to certain portion of a webpage. To specify the Mapping element, it is the same as figuring out the patterns that enclose each piece of information. The Mapping Wizard provides a GUI and tools to assist the users so that the manual efforts required for this step can be reduced as much as possible.

Mapping Rule Generation. Once all the information is confirmed, the Mapping Wizard generates the mapping rule according to the tree structure and all the corresponding properties. This process is fully automated. The Mapping Wizard validates the information that has been specified against the syntax defined in the WDM schema and produces the mapping rule.

Testing and Finalization. In this stage, the user tests the generated mapping rule to see whether it represents the correct logical structure and whether it locates all the required information. The testing is a trial-and-error process, similar to the process of debugging a program. The Mapping Wizard performs



Fig. 4. Main GUI of Mapping Wizard

the actual extraction using the generated mapping rule and returns either the retrieved information or error messages if any error occurs during the retrieval process. The user verifies whether the returned information is what he or she originally wants to extract.

The mapping rule is finalized once the user is satisfied with the information retrieved from the website. The final version can either be stored in some mapping rule repository or be delivered to users of the extraction agent for the extraction of information.

4 The Mapping Wizard

The current version of the *Mapping Wizard* is implemented using Visual C++ 6.0. This implementation is available on the Windows Operating System platform (Windows 98/ME/2000/XP). This section discusses the GUI of Mapping Wizard and how it helps users to quickly generate Mapping Rules.

4.1 Graphical User Interface

Figure 4 shows the main GUI of Mapping Wizard. The main components in this User Interface are the *Mini Web Browser*, the *Logical Tree Editor*, the *HTML Source Viewer*, and the *Property Dialog*.

Mini Web Browser. The Mini Web Browser on the right panel is a small Web browser for users to navigate the whole website. It has nearly all the functions of a normal Web browser. Most assistant tools rely on the Mini Web Browser to perform certain operations. For example, to use the Pattern Induction Tools, the user first highlights the target information on the Mini Web Browser and then activates the induction tool to discover the Mapping properties. The use of this browser instead of other external Web browsers is quite critical because the integrated features that rely on the Mini Web Browser can significantly improve the speed of the whole process.

HTML Source Viewer. The HTML Source Viewer displays the HTML source code of the webpage that the user is viewing in the Mini Web Browser. It is usually either located at the bottom of the main window or it pops up as a floating dialog window. Using the HTML Source Viewer enables users to view and analyse the HTML source directly. The assistant tools usually allow users to specify target information in two ways: on the Mini Web Browser or the HTML Source Viewer. The use of the HTML Source Viewer may yield higher accuracy in selecting the desired text string but it also requires relevant knowledge such as HTML and JavaScript.

Logical Tree Editor. The panel on the left is the Logical Tree Editor. The logical model of the website that the user is modeling is presented as a tree in this dialog. The editor allows users to perform normal tree manipulation on the logical tree, including insertion and deletion of tree nodes and modification of their properties.

Property Dialog. When the Property Dialog is opened, all the properties related to the current highlighted tree node in the Logical Tree Editor will be shown. Other than the use of the assistant tools to derive the properties, the Property Dialog is the only place that users are allowed to modify them.

4.2 Assistant Tools

The Mapping Wizard provides a set of assistant tools to help automate the process of generating a mapping rule. These tools are the critical components that make the Mapping Wizard practically useful. We believe that, in practice, it is not possible to have a single tool to fully automate the whole mapping rule generation process because the modeling of users' perception of the website's logical structure requires human intervention from time to time and the mapping from logical nodes to physical webpages is complex.

In the Mapping Wizard, rather than trying to provide a single tool, we build a set of tools that address different aspects of the problem. We identify the main difficulties and bottlenecks that tend to slow down the overall process

and develop assistant tools to accelerate them. In this way, the efficiency of the overall process is improved and the time required to generate a mapping rule of a given website is greatly reduced.

The assistant tools provided by the current implementation of the Mapping Wizard include Pattern Searching, HTML Source View Synchronizer, Pattern Induction Tools, Section Extraction Tool, and Structure Copier.

4.2.1 Pattern Searching Tool

Pattern Searching is similar to the “Search” function that most text editors provide. It allows the searching of certain text pattern within the HTML source and the Web browser view. Searching directly on the Mini Web Browser may be more helpful and straightforward to the users.

Searching with respect to current cursor position is an important feature for identifying the enclosing patterns of target information. When the user locates a text string as the target information and chooses the text string before that to be the BeginPattern, he or she would like to know whether this pattern appears before that position so as to make sure that the *Repeat* attribute of that pattern is specified correctly. The Pattern Searching tool can help in this case to search backward for the same pattern.

This tool may not seem important with the existence of the Pattern Induction Tools, which will be described later. However, the patterns derived by the Pattern Induction Tools may not be optimal in some cases. Users may want to modify and optimize the Mapping properties manually. In this case, Pattern Searching tools will be useful to test the manually modified patterns.

4.2.2 HTML Source View Synchronizer

When users locate information of interest, they usually browse through webpages using the Mini Web Browser. When the user has found the information on the webpage, he or she would like to see the corresponding HTML source codes of that portion. However, to locate the HTML source of certain part of a webpage is not straightforward, especially when the size of the webpage is large. The *HTML Source View Synchronizer* is implemented to accomplish this task of synchronizing the cursor position of the HTML source with the HTML View, and vice versa. The tool automatically highlights, in the HTML Source Viewer, the HTML source of the current selected portion on the Mini Web Browser, or the reverse if the current selected position is in the HTML Source. As shown in Figure 5, the user highlights the word “WORLD” and invokes the command. The corresponding HTML source is then highlighted by the Mapping Wizard, as indicated in Figure 6.



Fig. 5. Using the Source View Synchronizer

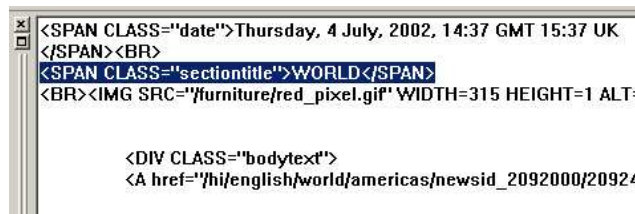


Fig. 6. Synchronized HTML Source

The synchronizer is almost always used when users try to locate the target information. The result of the synchronization has been proven to be quite accurate, with rare exceptions where complicated non-HTML techniques are used in webpages.

4.2.3 Pattern Induction Tools

The *Pattern Induction Tools* are the most useful ones among all the assistant tools provided. They are built based on the two tools described previously and combined with pattern induction algorithms.

These tools are developed to release users from manually figuring out the patterns that enclose the target information. Without these tools, after having identified the information, the user has to use the HTML Source View Synchronizer to locate the corresponding HTML source codes to analyze the patterns appearing before and after the target text string, and to use the Pattern Searching tool to confirm that the starting and ending patterns are unique, and to finally record these patterns in the Mapping properties of the logical tree node. All these steps can be done automatically with one click using the Pattern Induction Tools.

There are two types of target information that these Pattern Induction Tools deal with. Elements like Section, Region and Record only need to identify

the approximate area that contains the information of interest. They do not require their Mapping properties to enclose the exact boundary of the area, so long as all the interested information are enclosed. On the other hand, an Item element exactly locates a piece of information. It requires the Mapping properties to locate that information precisely. Therefore, the starting pattern and ending pattern must enclose the exact text string.

The current implementation of Mapping Wizard provides three Pattern Induction Tools: “Extract Region”, “Extract Record”, and “Extract Item”. The first two are for the first type where only a rough area is required, while the third one is for the second type.

Rough Area Pattern Induction For the first two tools, Mapping Wizard will detect what portion on the Mini Web Browser is highlighted and invoke the HTML Source View Synchronizer internally to determine the corresponding HTML source. Once the HTML source is found, the tools will apply an algorithm to derive the starting pattern and ending pattern that enclose this part of the HTML source. The algorithms used for inducing starting and ending patterns are symmetric. Here, we only discuss the algorithm for starting pattern.

The algorithm involves two repeating steps: *generating* a candidate and *testing* the generated candidate. It uses a trial and error approach to generate and test each candidate until a suitable candidate is found.

To generate candidates, the algorithm starts from the left of the target text string, and looks for the first complete HTML tag. For example, if the whole HTML source is

```
<TABLE><TR><TD>it's <FONT color=red>something interesting</FONT></TD></TR><TABLE>
```

and the target text string is

```
<FONT color=red>something interesting</FONT>
```

then the first candidate generated will be “<TD>”. To generate the second candidate, the starting position of the first candidate is expanded to the left until another complete HTML tag is found, which produces “<TR><TD>” in this example. The same rule applies for the rest of the candidates.

Once a candidate is found, it will be tested with two criteria: the length of the candidate pattern and the frequency that it appears before. These two criteria are chosen to ensure that the derived pattern is stable and flexible to the possible changes in the webpage in future. The more frequent that this pattern appears in the HTML page, the more likely that the number of times that it appears might change. Once the number of times changes, the Mapping

no longer locates the correct information. The same applies to the length of the candidate: the shorter the pattern, the more likely that it has a higher frequency of appearance. Therefore, to have a good induction result, suitable threshold values are required.

To test the first candidate, the algorithm calculates its length and uses the Pattern Searching tool to find out the number of times that this pattern “<TD>” appears, which is once. The testing is done by comparing these two number with the predefined thresholds. The values of the thresholds are determined empirically. In the implementation of the Mapping Wizard, we consider a candidate is suitable if it has a length longer than 5 and frequency less than 10. If a candidate is tested as suitable, it is taken as the derived value for the pattern and the algorithm terminates. Otherwise, the algorithm continues to generate the next candidate and test it.

In this example, the final patterns derived are “<TR><TD>” for BeginPattern and “</TD></TR>” for EndPattern. The text located by these patterns is

```
it's <FONT color=red>something interesting</FONT>
```

This string is not exactly the same as the one that we want. But since it contains the target information and we only require a rough area, the pattern induction result is acceptable.

This algorithm is relatively simple as compared with those used in similar systems, such as STALKER [23]. The extraction language used in STALKER is more complex and its training algorithm requires more than one training example in order to induce accurate patterns. Since in the context of the Mapping Wizard, the number of training examples is always one, we prefer a simpler extraction language and simpler training algorithm so as to achieve a higher accuracy of pattern induction. Nevertheless, as we mentioned earlier, a more complex extraction language, such as STALKER, can always be plugged into the WICCAP Data Modal and the Mapping Wizard, so long as the implementation of the extraction module and the training module is available.

Exact Area Pattern Induction For the “Extract Item” tool, patterns that enclose exactly the target information are required. The first step for determining the corresponding HTML source is the same as the Rough Area Pattern Induction. A similar algorithm is also applied to derive the patterns.

The algorithm differs from the first one only in the selection of the first candidate. Since exact enclosing patterns are required, for starting pattern, any candidate must end exact at the last character next to the left of the target text string. Therefore the first candidate is selected from the left of the target string to the first ‘<’ character. The generation of second candidate and so on is the same as the first algorithm.

In the same example above, the final patterns derived are “<TD>it’s” for BeginPattern and “</TD></TR>” for EndPattern. The text string located by these patterns now is exactly the same as the target text.

Exact Area Pattern Induction is particularly useful for deriving Mapping properties for *Item* elements. For example, to derive the patterns for an Item with type Link in the following text string

```
<A HREF="http://cais.ntu.edu.sg:8080/">
```

The final patterns induced are “” for EndPattern. The text string located by these patterns is “http://cais.ntu.edu.sg:8080/”, which is exactly the link that we want to extract.

As pointed out previously, these assistant tools can be used both on the Mini Web Browser and the HTML Source Viewer. If the user highlights on the Mini Web Browser, a synchronization between the HTML view and source is performed internally. Due to the highlighting mechanism implemented in the Microsoft Foundation Class (MFC), the user may not be able to select arbitrary portion of the HTML View on the Mini Web Browser. This is tolerable for Pattern Induction of rough area, which does not require an exact match and the user can always select a larger area. However, to achieve precise pattern induction for exact area, it is usually better to do the highlighting of the target information in the HTML Source Viewer.

4.2.4 Section Extraction Tool

When creating mapping rules, the first level of tree nodes is usually a list of links pointing to different pages. For example, in BBC Online News, the first level tree nodes are Sections such as “World”, “UK” and “Business”, each of which points to its own webpage. The links of these Sections exist in the homepage of the Website and are located close to each other within a small area.

With the tools described above, users have to highlight and let Mapping Wizard derive the patterns for each of these Sections, although they are all very similar. Since there could be quite a number of Sections, specifying the Mapping properties of them one by one is quite slow. The *Section Extraction Tool* is developed to handle this special case by attempting to figure out all the possible Section nodes in the same level and derive the Mapping properties for them in a single click.

The user highlights the area that contains the list of Sections on the Mini Web Browser and the tool finds out the HTML source using the HTML Source View

Synchronizer and applies suitable heuristics, based on common menu layout patterns, to identify the Sections. It then uses a similar induction algorithm to that of Exact Area Pattern Induction to derive the patterns that locate the links to the Sections' own webpages.

Normally, home pages or index pages of Web sites use some kind of menu to display a list of Sections. These menus usually make use of HTML TABLE or other tags to layout the menu items nicely. Based on this fact, we develop heuristics to locate each Section, depending on the type of tags that the Web site uses to construct its menu. The current implementation of Section Extraction Tool uses a simple heuristic, which blindly searches for the tag <A> and treats it as a Section. The patterns for locating the link are induced in a similar manner as illustrated in the example in last Section. The Name attribute of the Section node is extracted from the text enclosed by the tag <A> and . More complex heuristics that exploit the common menu layout structures will be explored in the future.

As an example, a website may have a menu like this

```
<TABLE>
  <TR><TD><A HREF="worldnews.htm">World</A></TD></TR>
  <TR><TD><A HREF="biznews.htm">Business</A></TD></TR>
</TABLE>
```

After applying the Section Extraction Tool, two Sections will be identified and inserted into the logical tree in the Logical Tree Editor automatically. These two Sections will have Name attribute of "World" and "Business" respectively. Their Mapping properties will be a Link containing a Locator that has a BeginPattern of "".

Our experiments showed that the simple heuristic used in our implementation performs quite well in the actual extraction. The tool successfully extracted Sections for all the tested websites. Even when deriving subsections under a Section, the tool is still able to recognize the Section names and patterns for the links correctly.

4.2.5 *Structure Copier*

After all the Sections nodes have been automatically inserted, the user has to insert the Region and Record and other child nodes under each Section and derive their Mapping properties. This works fine for a Web site with two or three Sections. But for a Web site that has 10 Sections in the first level, the same process of specifying Region, Record and Item under the first Section has to be repeated for another 9 times.

Fortunately, in most Websites, the structure of each Section is nearly identical, as they are normally generated by the same back-end engine with the same formatting styles. To make use of this fact to further accelerate the process, the Mapping Wizard provides another tool, the *Structure Copier*. The basic idea is to copy the structure of the constructed Section to other empty Section nodes, hoping that the same sub-tree hierarchy will fit into those Sections with little modification.

It should be pointed out that this assistant tool is in fact a special case of “Copy and Paste”. It can be seen as a combination of a single deep “Copy” and a batch “Paste”. A deep “Copy and Paste” means recursively copying all the descendants of the source node into the target node.

4.2.6 Combining All Tools

Using all the assistant tools together to create a mapping rule, the user first uses Section Extraction Tool to generate all the Sections, uses the Pattern Induction Tools to identify sub-tree nodes under the first Section, then activates the Structure Copier to copy the structure of the first Section to other Sections. After some modification on the rest of the Sections, a mapping rule is generated.

5 Results and Discussion

We have chosen four example sites (Figure 7) to conduct an evaluation of the Mapping Wizard. Most of these sites have been used for testing purposes by other similar systems. These four websites are chosen because they represent four very different categories of websites. Three users (with 1 to 2 years’ experience of HTML and some knowledge of XML) were asked to build a mapping rule for each website. For BBC Online News, the news title, link to the full article and the short description of the headlines in each section and subsection were modelled. For Amazon, we modelled the link, title, author and price of top 5 best sellers in the first 10 categories. For CIA, a selected set of 5 attributes of all countries starting with an “A” were wrapped. For DBLP, the pages of VLDB conferences, the names of authors, paper title, pages, and link of each paper from 1991 to 1999 were modelled.

Figure 8 summaries the key statistics of the experimental results. Users are required to record: (a) the number of nodes whose mapping properties are derived by the Pattern Induction Tool; (b) the number of sections that are extracted using the Section Extraction Tool; (c) the number of nodes that are created by using the Structure Copier; (d) the number of nodes that required

<i>Name</i>	<i>Website</i>	<i>Description</i>
BBC Online News	news.bbc.co.uk	News from BBC
Amazon	www.amazon.com	Amazon Books
CIA Fact Book	www.odci.gov/cia/publications/factbook/	Countries starting with A
DBLP	www.informatik.uni-trier.de/~ley/db/	VLDB Papers

Fig. 7. Test-sites used for Mapping Wizard

<i>Name</i>	<i>Nodes Derived</i>	<i>Section Extracted</i>	<i>Nodes Copied</i>	<i>Manually Modified</i>	<i>Time (mins)</i>	<i>Time Modeling</i>	<i>Time Mapping</i>
BBC	4	8	28	4 (9.1%)	27	11 (41%)	8 (30%)
Amazon	5	10	54	8 (10.4%)	41	17 (41%)	19 (46%)
CIA	11	20	238	7 (2.5%)	51	14 (27%)	29 (57%)
DBLP	12	9	96	6 (4.9%)	44	12 (27%)	20 (45%)

Fig. 8. Evaluation of Mapping Wizard

manual modification; (e) total time to construct the mapping rule; (f) time spent in constructing the logical model and (g) time spent in specifying the mapping details. Note that there is some amount of the time that was spent on waiting for testing result due to network speed and this time is the difference between (e) and the sum of (f) and (g).

From the result, we noticed that, except Amazon, all the three other websites required less than 10% of *manual modification* to the nodes automatically generated by the assistant tools. The amount of manual effort required is an important indicator of the efficiency of the Mapping Wizard and the degree of automation that it can offer. This figure largely depends on the efficiency of the two higher level tools Section Extraction Tools and Structure Copier. As we mentioned earlier, these two tools manipulate a group of nodes instead of a single node at a time. The more nodes they can generate, the higher the efficiency of the overall process, thus the higher the degree of automation. This is especially true for the Structure Copier, which may result in a large number of nodes copied. In the case of CIA Fact Book, as the page for each country has the identical structure, simple structure copying worked very well and had saved the users a lot of effort, which resulted in the lowest manual modification. The 10.4% manual effort required for Amazon was partly due to the fact that the users were required to model only 10 categories of the books. If all (35) categories were modelled, this figure will be significantly reduced (estimated 3.2%).

The total time required to construct a mapping rule for each website is within one hour. One of the most recent systems, Lixto [5], requires about half of the amount of time. However, the time required depends very much on the granularity of the information that the constructed wrapper can extract (i.e. the depth of the tree and the number of nodes in the tree). In addition, the WICCAP Data Model deals with the whole website while Lixto seems to focus on a single webpage in its experiment. To form a site-level view, similar amount

of time has to be spent on each webpage that the users are interested in and some additional time is also required to combine all the separate page view into an integrated site view. The total time would obviously be much larger. Therefore, we consider the Mapping Wizard a much more efficient tool than other similar systems.

We also saw a relatively consistent amount of time (from 11 to 17 minutes) spent on figuring out and constructing the logical skeleton of the websites. The time for specifying the mapping details varied depending on the number of nodes in the actual mapping rules. The CIA Fact Book required the longest time because there is a huge amount of information for each country, which leads to a large number of nodes. Thus, the total time needed is dependent on the granularity of the mapping rules constructed.

All users were able to obtain 100% accuracy for modeling the websites using the Mapping Wizard. No users had to manually edit the generated mapping rule's XML document.

6 Related Work

Web information extraction systems are also known as *wrapper generation systems*, which was traditionally divided into three groups: *manual wrapper generation*, *wrapper induction*, and *supervised wrapper generation*. This classification scheme has gradually become too simplistic with recent appearance of several interesting Web information extraction systems. A more detailed classification is discussed in [16].

Systems such as Tsimmis [11], Araneus [21] and Jedi [13] support only *manual wrapper generation*. These systems provide expressive and comprehensive languages for information extraction. The extraction languages in these systems are so complex that they have to be written manually by some experts users. There is no algorithm that helps derive the languages from examples or GUI support for the ease of writing the extraction rules. Thus, manual wrapper generation systems tend to be difficult to use.

Another category of systems, including Wien [14], SoftMealy [12] and STALKER [23], take the *machine learning* approach to learn the extraction rules by examples. They are termed *wrapper induction systems*, because algorithms have been developed to induce wrappers from a set of marked examples. These systems also provide visual support to allow easy selection and labelling of training examples. However, Wien is the pioneering work in this area and can not handle sources with nested structure and missing and varying-order attributes. SoftMealy requires seeing all possible special cases (missing at-

tributes or out of order attributes) in the training phase in order to induce a correct wrapper. STALKER provides a more expressive extraction language by allowing disjunction and multiple patterns for locating information (WDM only supports a single starting or ending pattern). All these wrapper induction systems have been traditionally seen as automatic wrapper generation systems. But before the automatic training process, they all require a phase of marking of examples which might be non-trivial. When the overhead of marking training examples overtake the advantage given by automatic training, it may not be worthy to use such systems. In addition, all these three systems, and most of other similar systems, only deal with a single webpage or a set of webpages with similar structure. They lack the ability to traverse along hyper-links; thus unable to extract information appearing in multiple inter-linked webpages.

Our work is most closely related to *supervised wrapper generation systems*. Systems in this category include DEByE [15], Lixto [5], NoDoSE [1], XWrap [18] and W4F [24]. A common feature of these systems is the GUI tools together with internal algorithms for extraction rules derivation that have been developed to support the generation of wrappers interactively. Although the wrapper generation process is not fully automatic, they attempt to reduce the required human intervention as much as possible.

DEByE allows users to specify a structure according to their perception of the information source. This is similar to WDM's logical view of website, although DEByE deals with information only at page-level. Unlike in the Mapping Wizard, this user-perceived structure is not specified explicitly but at the time when the users construct the examples using nested-table. The proposed data model and extraction language (called *oe-patterns*) only allow extraction patterns to exist in leaf nodes. This leads to the preference to a bottom-up extraction strategy. On the contrary, the WICCAP Data Model allows both internal nodes and leaf nodes to have their extraction patterns (the Mapping element). This makes a top-down extraction strategy more suitable and eliminates the need for an assembling phase for grouping instances of attributes together, which is required in DEByE.

Another very similar tool is the Lixto system. Lixto allows users to interactively build wrappers by creating *patterns* (tree nodes) in a hierarchical manner. The extraction rule associated with each pattern is a disjunction of one or more *filter*, each may consist of a conjunction of *conditions*. Such extraction language, called Elog, is as expressive as Monadic Datalog [10]. One advantage of Lixto is that although the internal Elog language is rather complex, the visual tools provided allow users to deal only with the GUI and the browser view of the HTML webpages, while not touching the underlying Elog rules. This is possible because a filter can be represented by an Elog rule and Lixto provides tools of very fine granularity that support the specification of

every single filter and condition. The nested structure of patterns implicitly forms certain logical structure as the users compose the pattern hierarchy. This is again different from WDM where the logical structure is explicitly specified. The Elog language includes a predicate *getDocument(S,X)* that allows crawling across Web documents. This potentially gives Lixto the ability to model and extract data from multiple webpages, although the tools do not explicitly provide such support. The extracted results in Lixto can be manipulated by translating them into XML. Similarly, in the WICCAP system, the second layer that performs the extraction is also capable of performing (more involved) post-processing tasks such as filtering and consolidation [17].

Although DEByE and Lixto both have an expressive user interface for creating wrappers, the tools available are mostly for wrapper generation operations that are of the lowest granularity, i.e. for specifying extraction patterns for each individual node. Such kinds of tools are too low level when dealing with large scale websites. The wrapper generation process will be tedious and time-consuming, although the derivation of patterns for each individual node is efficient. On the contrary, the Mapping Wizard provides higher level tools such as Section Extraction Tool and Structure Copier that manipulate several nodes and even sub-trees. Although such high level assistant tools may not achieve 100% accuracy, they can greatly shorten the time and reduce the manual effort involved, as shown in our experiments.

NoDoSE adopts a similar approach as the Mapping Wizard to generate wrappers. It uses a similar top-down strategy to decompose a document into a hierarchical structure. However, in NoDoSE, the whole documents must be decomposed completely, while in the Mapping Wizard the logical structure is constructed independently of the webpages and only representative portions of the pages need to be marked. Moreover, NoDoSE is unable to follow links in HTML pages, hence unable to model a whole website. Nevertheless, NoDoSE is designed to handle a broader category of information sources, namely plain text. This is a feature that most Web information extraction systems lack.

A different approach based on HTML DOM tree is used in XWrap and W4F, where the users see the HTML DOM tree instead of the HTML source when building the wrappers. In XWrap, users indicate the portions to be extracted by browsing the DOM tree and selecting relevant nodes. A set of pre-defined heuristics can be applied after the users have selected some portion to derive the actual extraction rules. The ability for the users to specify their views of the logical structure is limited because the hierarchical structure extraction is performed based on predefined XML templates that contains instructions on deriving the structure.

W4F offers an expressive extraction language, called HEL, that can be used to perform extraction based on the HTML DOM tree. However, the users are

required to write the extraction rules in HEL manually. Although W4F offers a set of GUI toolkits to help users accomplish this goal, there is no algorithm for automatically deriving the extraction rules. From this point of view, W4F is closer to the category of manual wrapper generation, rather than supervised wrapper generation.

One system that does not fall into the 3-group classification is the work by the BYU Data Extraction Research Group (DEG) [9]. It is based on an ontology-based approach that makes use of a semantic data model, called *application ontology*, which contains both a narrow ontology about the domain of interest and the keywords and constraints for the extraction of individual data items. Once the application ontology of a particular domain has been produced, the extraction of data from any webpage from the same domain is fully automatic, which is what all the systems mentioned above, including WICCAP, can not achieve. However, the construction of an application ontology is time-consuming and requires an expert who is familiar with both the syntax of the ontology language and the target domain. Although the structure exhibited by the application ontology reflects the logical organization of the data on the target page, it is rather narrow and usually has only one or two levels. In addition, it only deals with a single page. Therefore, the ability to model logical views of websites is considerably limited.

Two other systems that go further in terms of the degree of automation are RoadRunner [8] and the ExAlg algorithm [2]. The whole wrapper derivation process in these two systems are fully automated. They do not require any human intervention except for supplying target pages with similar structure (pages that are generated from the same template). This is a distinct feature comparing with all other wrapper generation systems. However, one disadvantage of such kind of automatic wrapper derivation systems is that they extract the syntactic structure, instead of the semantic one. The users have to perform some post-processing steps in order to get the real logical view that they want. RoadRunner and ExAlg are also limited to deriving wrappers at page-level.

Among all these related systems, only few (DEByE, Lixto and Wien) offer the similar functionality as the Mapping Wizard to allow users to directly operate on the browser view of webpages instead of the HTML sources. Being able to operate directly on the browser view is critical to a supervised wrapper generation system because it does not require the user to have in-depth knowledge about HTML and JavaScript.

Most systems, including DEByE, Wien, XWrap, W4F, the BYU DEG work and RoadRunner, deal with only a single webpage or a set of webpages with similar structure. This greatly limits the ability to modeling a website as a whole. Although it is possible to aggregate several page views, produced by

these systems, to form a higher level site view, the efforts required to compose the site view out of several page views may not be trivial, especially when this has to be done manually without any support of visual tools. On the contrary, the WICCAP Data Model allows extraction of data from multiple inter-linked webpages by explicitly incorporating the concept of a hyper-link. This gives users the flexibility in deciding the scale of the information sources that they want to model, i.e. the users can either choose model the whole website, a single webpage, or just a segment of a page.

The Araneus data model (ADM) used in the Araneus project is able to model an entire website. It adopts a variant of the ODMG model [7] and explicitly introduces the concept of a *page scheme* to model pages of similar structure. The attributes of a page scheme can be rather complex and may point to other page schemes. Although the ADM is able to model a wider class of websites (it uses a directed graph rather than a tree), the use of page in the model makes the resulting models close to the physical structure of websites. In the WICCAP Data Model, the concept of a webpage is eliminated and the data on a website is modelled solely based on their logical structure. The user may choose to represent a page with some WDM element; but most of the time, WDM elements will be used to represent just a fragment of a page. Unlike in ADM, where the user sees the hyperlinks when they look at the data model, the WICCAP Data Model hides the mapping details in the Mapping related elements. When showing the logical tree with only the WDM elements for logical modeling purpose, the physical structure of the website is completely hidden from the user.

7 Conclusions

In this paper, we introduce a new data model (WICCAP Data Model) for mapping websites from their physical structures to logical views. This data model is capable of describing the hidden logical structure of multiple inter-linked webpages. We believe that the ability to model at site-level into the data model is important for building logical views of websites. Our proposed model also achieves a complete decoupling of the logical view from the physical structure of websites by defining a dedicated group of elements for mapping purpose.

The Mapping Wizard has been implemented as a software tool to automate the process of generating a data model. Since a fully automatic algorithm is not possible, several tools have been developed to address various bottlenecks in the generation process. Assistant tools such as Section Extraction Tool and Structure Copier offer higher degree of automation by operating on a group of nodes at the same time. By combining all the tools, the overall process of creating logical views of websites can be significantly accelerated.

A A Mapping Rule of BBC Online News

```
<!-- WiccapNewsBBC.xml -->
<?xml version="1.0" encoding="UTF-8"?>
  <Wiccap Name="BBC" Group="News"
    xmlns:xsi="http://www.w3.org/2000/10/XMLSchema-instance"
    xsi:noNamespaceSchemaLocation="WiccapNews.xsd">
    <Mapping><Link Type="Static">http://news.bbc.co.uk</Link></Mapping>
    <Section Name="World">
      <Mapping>
        <Link Type="Dynamic"><Locator Type="ByPattern">
          <LocatorPattern>
            <BeginPattern>
              <![CDATA[<TD CLASS="rootsection" BGCOLOR="#FFFFFF"><A HREF="]]>
            </BeginPattern>
            <EndPattern><![CDATA[" CLASS="index">]]></EndPattern>
          </LocatorPattern>
        </Locator></Link>
      </Mapping>
      <Region Name="ArticleList">
        <Mapping><Locator Type="ByPattern">
          <LocatorPattern>
            <BeginPattern><![CDATA[<SPAN CLASS="sectiontitle">]]></BeginPattern>
            <EndPattern><![CDATA[<SCRIPT LANGUAGE=]]></EndPattern>
          </LocatorPattern>
        </Locator></Mapping>
        <Record Name="Article" NumOfItem="3">
          <Mapping><Locator Type="ByPattern"><LocatorPattern>
            <BeginPattern><![CDATA[<DIV CLASS="]]></BeginPattern>
            <EndPattern/>
          </LocatorPattern></Locator></Mapping>
          <Item Type="Link" TagFilter="Off" Description="This is the link to the news">
            <Mapping><Locator Type="ByPattern"><LocatorPattern>
              <BeginPattern><![CDATA[<a href="]]></BeginPattern>
              <EndPattern><![CDATA[">]]></EndPattern>
            </LocatorPattern></Locator></Mapping>
          </Item>
          <Item Type="Title" Description="This is the title of the news">
            <Mapping><Locator Type="ByPattern"><LocatorPattern>
              <BeginPattern><![CDATA[<B class="h***]]></BeginPattern>
              <EndPattern><![CDATA[</B>]]></EndPattern>
            </LocatorPattern></Locator></Mapping>
          </Item>
          <Item Type="Description" Description="This is the description of the news">
            <Mapping><Locator Type="ByPattern"><LocatorPattern>
              <BeginPattern><![CDATA[</A>]]></BeginPattern>
              <EndPattern><![CDATA[<BR>]]></EndPattern>
            </LocatorPattern></Locator></Mapping>
          </Item>
        </Record>
      </Region>
      <Section Name="Asia-Pacific">
        : : :
      </Section>
      <Section Name="Americas">
        : : :
      </Section>
      <Section Name="Business">
        : : :
    </Wiccap>
```

References

- [1] B. Adelberg, NoDoSE - A Tool for Semi-Automatically Extracting Semi-Structured Data from Text Documents, in: Proceedings of 1998 ACM SIGMOD International Conference on Management of Data, Seattle, Washington, USA, 1998, pp. 283–294.
- [2] A. Arasu, H. Garcia-Molina, Extracting Structured Data from Web Pages, in: Proceedings of ACM SIGMOD International Conference on Management of Data (SIGMOD 2003), ACM Press, San Diego, California, USA, 2003, pp. 337–348.
- [3] N. Ashish, C. A. Knoblock, Semi-Automatic Wrapper Generation for Internet Information Sources, ACM SIGMOD Record 26 (4) (1997) 8–15.
- [4] P. Atzeni, G. Mecca, P. Merialdo, To Weave the Web, in: Proceedings of International Conference on Very Large Data Bases (VLDB 97), Athens, Greece, 1997, pp. 206–215.
- [5] R. Baumgartner, S. Flesca, G. Gottlob, Visual Web Information Extraction with Lixto, in: Proceedings of 27th International Conference on Very Large Data Bases (VLDB 2001), Roma, Italy, 2001, pp. 119–128.
- [6] BBC Online News, <http://news.bbc.co.uk/> (2002).
- [7] R. G. G. Cattell, T. Atwood (Eds.), The Object Database Standard: ODMG-93: Release 1.2, Morgan Kaufmann, 1996.
- [8] V. Crescenzi, G. Mecca, P. Merialdo, RoadRunner: Towards Automatic Data Extraction from Large Web Sites, in: Proceedings of 27th International Conference on Very Large Data Bases (VLDB 2001), Roma, Italy, 2001, pp. 109–118.
- [9] D. W. Embley, D. M. Campbell, Y. S. Jiang, S. W. Liddle, D. W. Lonsdale, Y.-K. Ng, R. D. Smith, Conceptual-model-based data extraction from multiple-record Web pages, Data & Knowledge Engineering 31 (3) (1999) 227–251.
- [10] G. Gottlob, C. Koch, Monadic Datalog and the Expressive Power of Languages for Web Information Extraction, in: Proceedings of 21st ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems (PODS 2002), Madison, Wisconsin, USA, 2002, pp. 17–28.
- [11] J. Hammer, H. García-Molina, J. Cho, A. Crespo, R. Aranha, Extracting Semistructured Information from the Web, in: Proceedings of the Workshop on Management of Semistructured Data, Tucson, Arizona, USA, 1997, pp. 18–25.
- [12] C.-N. Hsu, M.-T. Dung, Generating Finite-State Transducers for semistructured Data Extraction From the Web, Information Systems 23 (8) (1998) 521–538.

- [13] G. Huck, P. Fankhauser, K. Aberer, E. J. Neuhold, Jedi: Extracting and Synthesizing Information from the Web, in: Proceedings of the 3rd IFCIS International Conference on Cooperative Information Systems (CoopIS 98), New York City, New York, USA, 1998, pp. 32–43.
- [14] N. Kushmerick, Wrapper induction: Efficiency and expressiveness, in: AAAI-98 Workshop on AI and Information Integration, Madison, Wisconsin, 1998, pp. 15–68.
- [15] A. H. F. Laender, B. A. Ribeiro-Neto, A. S. da Silva, DEByE – Data Extraction By Example, *Data & Knowledge Engineering* 40 (2) (2002) 121–154.
- [16] A. H. F. Laender, B. A. Ribeiro-Neto, A. S. da Silva, J. S. Teixeira, A Brief Survey of Web Data Extraction Tools, *ACM SIGMOD Record* 31 (2) (2002) 84–93.
- [17] F. Li, Z. Liu, Y. Huang, W. K. Ng, An Information Concierge for the Web, in: Proceedings of the First International Workshop on Internet Bots: Systems and Applications (INBOSA 2001), in conjunction with the 12th International Conference on Database and Expert System Applications (DEXA 2001), Munich, Germany, 2001, pp. 672–676.
- [18] L. Liu, C. Pu, W. Han, XWRAP: An XML-Enabled Wrapper Construction System for Web Information Sources, in: Proceedings of the 16th International Conference on Data Engineering (ICDE 2000), San Diego, California, USA, 2000, pp. 611–621.
- [19] Z. Liu, F. Li, W. K. Ng, Wiccap Data Model: Mapping Physical Websites to Logical Views, in: Proceedings of the 21st International Conference on Conceptual Modelling (ER 2002), Tampere, Finland, 2002, pp. 120–134.
- [20] Z. Liu, F. Li, W. K. Ng, E.-P. Lim, A Visual Tool for Building Logical Data Models of Websites, in: Fourth ACM CIKM International Workshop on Web Information and Data Management (WIDM 2002), McLean, Virginia, USA, 2002, pp. 92–95.
- [21] G. Mecca, P. Atzeni, Cut and Paste, *Journal of Computer and System Sciences* 58 (3) (1999) 453–482.
- [22] I. Muslea, Extraction Patterns for Information Extraction Tasks: A Survey, in: Proceedings of Workshop on Machine Learning for Information Extraction, Orlando, Florida, USA, 1999, pp. 1–6.
- [23] I. Muslea, S. Minton, C. Knoblock, Hierarchical Wrapper Induction for Semistructured Information Sources, *Autonomous Agents and Multi-Agent Systems* 4 (1–2) (2001) 93–114.
- [24] A. Sahuguet, F. Azavant, Building intelligent Web applications using lightweight wrappers, *Data & Knowledge Engineering* 36 (3) (2001) 283–316.
- [25] XML Schema, <http://www.w3.org/XML/Schema/> (2003).