Knowledge-Based Approach for Automating Web Publishing from Databases

Zhangxi Lin
Department of ISQS
Texas Tech University
Lubbock, Texas 79409-2101

Matti Hämäläinen
Codeonline Ltd.
Ukonvaaja 2
02130 Espoo, Finland

Andrew B. Whinston
The Center for Research in Electronic Commerce
Graduate School of Business
The University of Texas at Austin
Austin, Texas 78712-1175

I. Introduction

The exponential growth rate of the Internet has provided an ever-increasing virtual information space for research, education, and business. In particular, the user-friendly interfaces, multimedia accessibility, client–server computing architecture, and extendibility of Web servers and clients have been exerting more and more influence on the way people and organizations do things. Since the opening of the Internet for commercial use it has been rapidly becoming the most important medium for online publishing of timely information [1, 2].

The power of the Web comes from a simple but powerful document encoding language, HyperText Markup Language (HTML), a set of transfer protocols that support the access to distributed documents, and browsers that are capable of presenting multimedia documents. The tools used for producing HTML documents range from plain text editors to Web publishing systems, and the Web tool market continues to grow rapidly. Although the early authoring tools were meant for manual tagging of HTML elements, the more recent tools for routine HTML authoring allow what-you-see-is-what-you-get (WYSIWYG) editing (e.g. Netscape Composer or MicroSoft FrontPage). The authoring tools are appropriate to support interactive authoring of HTML-based documents, where the generation of documents and maintenance of their interrelationships is done manually. For turning conventional documents into HTML, the converters or filters have been developed for most common document types. These include the HTML conversion facilities that are built into office packages, such as Microsoft Office (see http://www.microsoft.com). They can do much of the conversion automatically based on the logical styles defined in the original source document (e.g., a Word file). They can handle many HTML elements, such as tables, and convert figures to GIF or JPEG files. In addition to the single-file conversion, batch-type converters (e.g., HTML Transit from InfoAccess, see http://www.infoaccess.com) are also available for processing large numbers of documents. They support definition of templates of documents for standardization, create indices and table of contents, and support division into multiple HTML files. These are useful for batch processing of large volumes or long documents and are good for both HTML language novices and professionals. The next category includes the Web site publishing tools, such as Microsoft FrontPage (see http://www.microsoft.com/products/prodref/571_ov.htm) and NetFusion Objects (see http://www.netobjects.com/) that have facilities for managing large collections of Web documents. Typically they have templates for standardizing document layout, support WYSIWYG editing of HTML, configuration management (i.e., when a file name changes, all links are updated automatically), and updating of the Web documents on remote servers.

However, these authoring and publishing tools are not adequate for publishing from large volumes of rapidly changing information sources. Database publishing, where the Web pages are created dynamically from database contents and are based on diversified user requests, is a rapidly growing method for producing Web pages.

In this chapter we introduce a knowledge-based scheme for facilitating production and maintenance of HTML-based documentation. Because a Web application project may include various different tasks depending on application type, document size, user task classification, network access quality, etc., we restrict our approach to the applications that require certain data analysis based on constantly updated databases as well as reusable Web page building components. We claim that certain kinds of routine documentation based on a regularly updated database or files can be much more efficiently produced and kept up-to-date by applying the proposed approach. In fact, much progress has been achieved in text document generation. Knowledge-based methodology has been applied to generate text reports for market studies [3], accounting analysis [4], financial analysis [5], commodity price analysis [6], etc. The application of knowledge-based methodology now prevails mainly in Web information searching [7]. We started the efforts in knowledge-based HTML generation in 1995 [8, 9]. The process of generating HTML documents involves not only content editing, but also maintenance of links among multiple files and file types, with special markup tags; a well-designed Web site requires elaborate planning of both the logical structure and the layout structure. Recent research in Web site design has enriched its systematic development approaches. For example, [10][1] has presented a comprehensive analysis on how to design an information-abundant Web site. We will show that knowledge-based techniques could be utilized in this task to complement those used in conventional text generation.

[1]Several relevant papers can be found in the same issue of *International Journal of Human–Computer Studies* (1997, Vol.47, No.1).

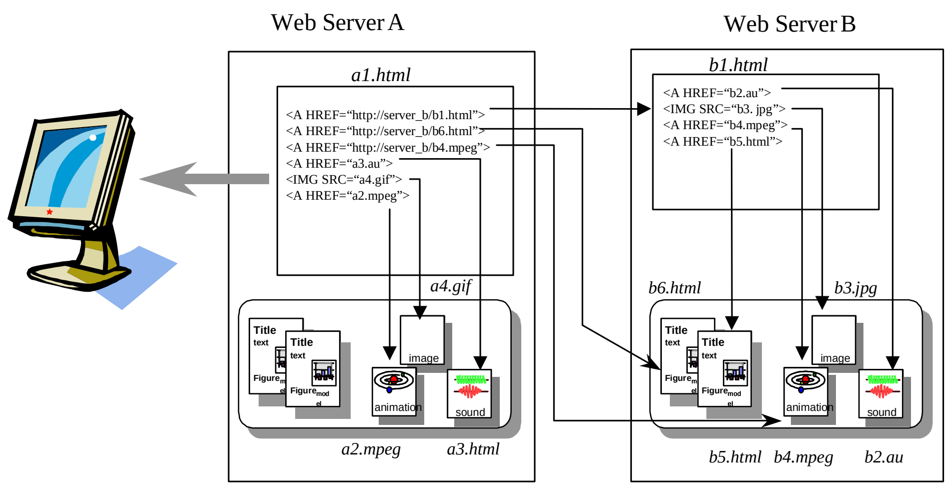
II. Automating HTML page generation

A. HTML Document Generation

An HTML encoded file has the ability to link regions of text or images to other documents that may reside on other Web servers. The possible access methods for the linked objects include *http, ftp, gopher, wais, news,* and *telnet*. The data types of documents vary among HTML-tagged text, plain text, PostScript file, image, animation, audio, video, etc. Actually almost any user-specified type can be accessed and viewed when properly defined in the configuration profile of the browser. The processing mechanism for the basic types, such as HTML files, and GIF and JPEG images, is built into the Web browsers and is well supported.

Our main focus in this chapter is the logical relationship among documents. One of the major tasks specific to HTML document generation, but not to normal text preparation, is the planning of the links within document clusters.

Figure 1 shows a simple model of HTML documents stored in a distributed environment with hypermedia processing features. Even for a small or medium-sized HTML documentation task, usually several files are referred to and multiple access to several servers is necessary. Because document planning and document component managing take more time, HTML documents are much more complicated and less productive to design, produce, and maintain than a conventional word processing task.



```
<A HREF="http://server_b/b1.html">
<A HREF="http://server_b/b6.html">
<A HREF="http://server_b/b4.mpeg">
<A HREF="a3.au">
<IMG SRC="a4.gif">
<A HREF="a2.mpeg">
```

```
<A HREF="b2.au">
<IMG SRC="b3. jpg">
<A HREF="b4.mpeg">
<A HREF="b5.html">
```

Figure 1 HTML document links among servers on the Internet.

Figure 2 illustrates the structural parts in an HTML application design. Source information[2] materials are the basis of the documentation. These materials are to be converted into HTML components (e.g., HTML, GIF, JPEG, MPEG, AU files) if they are not in the proper formats. The appearance of a document component is relevant to the user interface. This will in turn affect the design of each document component. While producing and assembling document components, the designer is also configuring the hyperlink structure. This structure puts each document component in the appropriate layout location. For a practical application these tasks are usually carried out regressively.

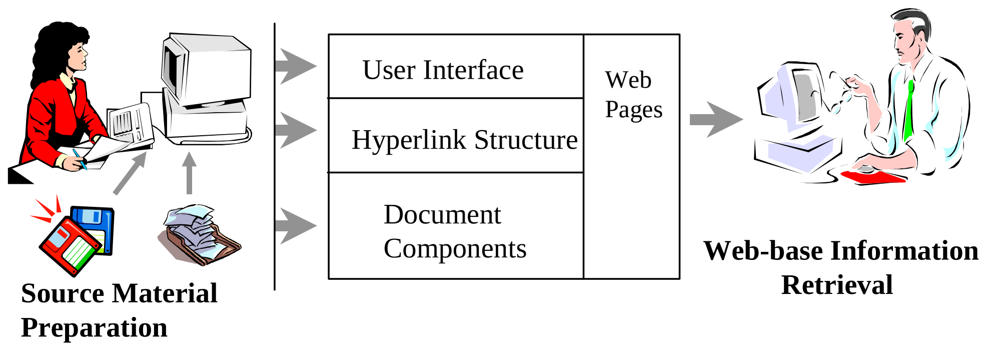


Figure 2 The structural parts of HTML application design.

When publishing electronic documents, such as journals, newsletters, and surveys on the Web, the HTML documents are created and maintained regularly, whereas much of the content is based on periodically updated information in certain formats (which are based on the Internet). Even though a database may be available on the Internet, it is beneficial to produce new HTML components by summarizing newly updated data or by presenting users with a more friendly appearance of the database with multimedia techniques. Actually, Web interfaces to databases are already common and all the major database vendors (e.g., Microsoft, Oracle and Sybase) offer Web interfaces and complete tool sets for robust and secure interfacing of Web and databases. For NT based Web servers the Microsoft's Open Database Connectivity (ODBC) has

[2]We use the notion of "source information" to denote the various periodically updated sources for automated Web publishing, such as databases, reporting programs and online services accessible from either local or remote servers.

become very popular, while Java Database Connectivity (JDBC) is emerging as a more versatile tier to provide an excellent interface for Java applications. To improve database access efficiency, a number of application programming interfaces (APIs) for Web-based database applications, such as Netscape's Netscape Server API (NSAPI) and Microsoft's Internet Server Application Program Interface (ISAPI), have emerged to replace the conventional Common Gateway Interface (CGI) technology.

Although these provide technical capabilities for extending the conventional static Web publishing schemes, we suggest, that the cost-efficiency and quality of Web publishing can be improved via the integration of DBMS and knowledge-based techniques:

- The skills of routine information analysis and HTML document generation can be extracted as expertise to be stored and utilized in a form of a knowledge base;
- The repeatedly used documents and their revisions can be managed with database techniques; and this database can be coupled to a knowledge base for advanced usage [11].
- The multiple-tier client–server computing model makes it easier to plug in a knowledge-based layer between databases and application kernel.

B. Principles of the Knowledge-based HTML Document Generator

The knowledge-based HTML document generator (KHDG) is rooted in two assumptions:

- There is a need for routine publication of information generated by an analysis of regularly updated information sources.
- There is a collection of reusable document components in a repository, potentially distributed across several Web servers.

The need for dynamic HTML page generation arises primarily from the following two cases:

- When the source information has been updated, new versions of HTML documents need to be produced, in order to provide the latest up-to-date views of the information via the Web.
- The information retrieved from Web-based databases is subject to change in accordance with the user queries.

The outcome of the final HTML documents includes the core information from the updated source information, combined with the complementary information from the document components in the repository. Therefore, there must be an integrated authoring facility in addition to the database or files used as information sources.

The composition process consists of three steps similar to the manual approach:

- *Source information analysis* — First, KHDG must scan the source information that has just been updated, to obtain the contents to be published. This results from successive reasoning steps. A goal for a desired HTML page construction must be set to guide the reasoning process.
- *Document planning* — The hypertext documents must be well planned before being generated. Specifically, document hyperlink planning is crucial so as to optimize the document application structure.
- *HTML file generation* — In this phase, HTML-tagged documents are produced in accordance with the requested typeset formats. The output will be directed to a set of HTML files based on the structure conceived in the planning stage.

III. Representing the Knowledge for KHDG

Figure 3 depicts the internal structure of KHDG. The three steps for HTML document generation are assigned to the three subsystems in KHDG*: data analyzer* (DAN), *document planner* (DOP) and *HTML file generator* (HFG). The knowledge base is implemented by four facilities: *source material repository* (SMR), *document catalog base* (DCB), *HTML documentation framework* (HDF), and *analyzing & authoring expertise* (AAE).

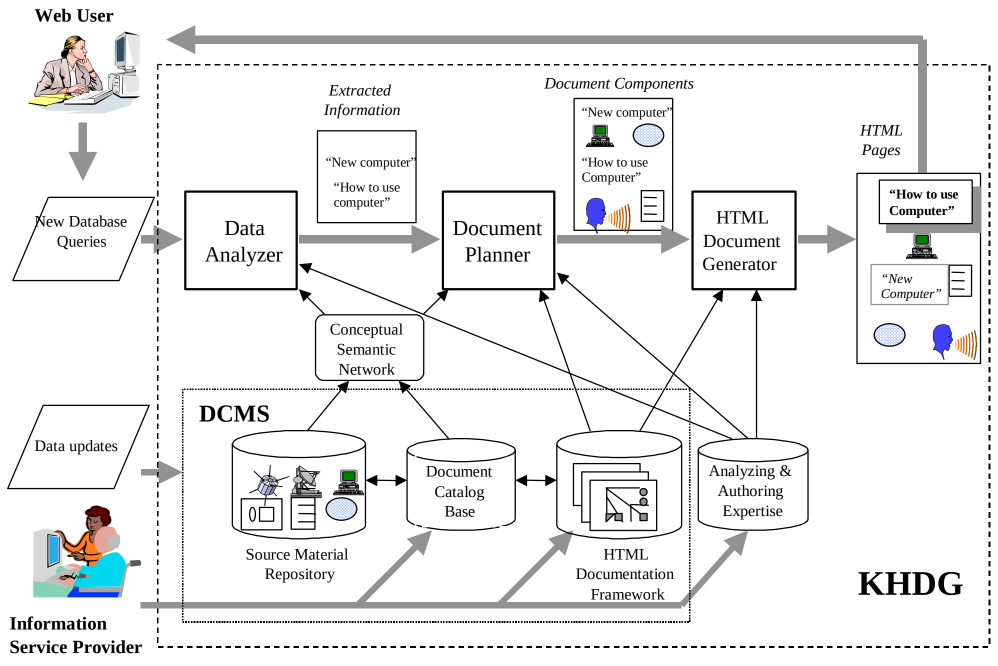


Figure 3 Internal structure of the KHDG system.

A. Managing Constructive Document Components

The primary architecture of HTML document collections is convenient for browsing hypertext materials, but is not suitable for data management, because the document material repository is actually a collection of regular files. The relationship among them relies heavily on the links from some regions inside the files to other files. Therefore, the *document component management system* (DCMS) is required to integrate document components in a proper architecture for more effective manipulation. Each document component to be stored in DCMS will have a description that contains metainformation about the component. This information results from the information analysis phase and is used for both component management and component retrieval. Typical metainformation fields are creator, creation date, expiration date, type of material, and descriptive keywords.

The metainformation will be used for searching for appropriate candidate components for a document to be composed. This information will provide the support both for finding the interrelated components and for the automatic or semiautomatic layout of document components on the pages or frames. In the automatic construction case the system will automatically pick the components, match them with potentially suitable templates, and fill in the templates with the components. In the semiautomatic case the idea is to provide HTML scripting suggestions for the composer of the document; so he or she can pick the components that fit the profile (query) and can easily organize them as pages using a predefined set of document templates (typically represented as frame sets) for the HTML pages.

The composition system will produce a list of files (list of URLs) and, by using templates (page definitions), help to rapidly compose the publication as a set of interconnected Web documents with appropriate layout and functionality (e.g., navigation buttons) added. The database provides a mechanism to store the relationships for elements with certain layout information (e.g., links to templates).

The document components maintained in DCMS are classified into major types, such as text, image, sound, computerized diagram, computerized animation, and video. DCMS must be capable of handling these types of files and must possess the basic features that a normal DBMS has, such as data integrity, data consistency, and system security.

KHDG employs DCB, SMR, and HDF to implement DCMS' functions (Figure 4). SMR is virtually a collection of HTML components distributed on several Web servers, which are mapped to records in DCB; DCB maintains the catalogs of SMR components. They are paired at four levels: Subject Page Catalog and Subject Page Repository (SPC–SPR), Topic Catalog and Topic Repository (TOC–TOR), Fragment Catalog and Fragment Repository (FRC–FRR), and Elementary Unit Catalog and Elementary Unit Repository (EUC–EUR). The first two levels are used for manipulating HTML files. SPC–SPR is just for complete HTML files that can be directly accessed by users.

The components in TOR are author-oriented HTML files which are used as functional HTML components for documents in SPR. The internal structure of TOC can comprise any sublevels depending on needs. For example, an HTML file for a chapter is indexed in TOC. This chapter has several pointers to other topics while these topics may consist of further detailed subtopics in HTML files. Some topic files need more than one component to build up, so that they may go several levels deeper. This multilevel structure provides TOC more flexible capability to express the logical structure of a physical HTML document.

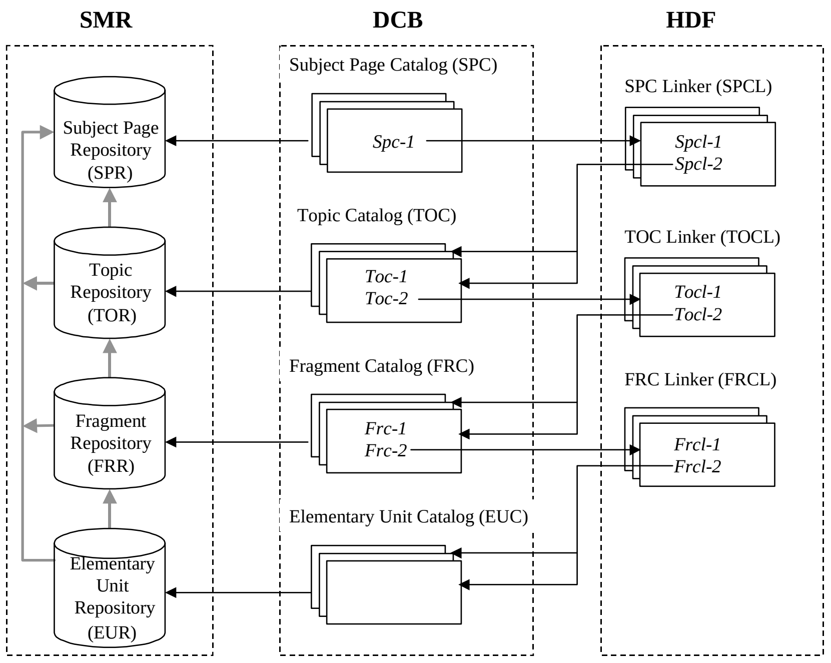


Figure 4 The infrastructure of DCMS.

FRR is a repository for fragmental HTML components that are construction blocks for functional components in TOR. It is abstracted by FRC in a multileveled structure. The difference between FRC and the two upper level catalogs is that components in FRR, the counterpart of FRC, can not have a URL, because they are only constructive fragments for regular HTML documents.

EUC is for non-text atomic components, the multimedia data files typically with extension .gif, .mpeg, .au, .jpg, or .bmp.

HDF manages HTML construction templates and the structure of dynamic HTML documents. Both types of frameworks are represented in the links among document components, which are stored in three linker files: SPCL, TOCL, and FRCL. Working with four catalog files, the three linker files establish the whole relationship among document components.

With a unified DCMS, it is convenient for KHDG to store, retrieve, add, modify, and delete document components distributed on several Web servers. More important, all the views and versions of publications can be managed with DCMS as organized document components.

B. Representing Authoring Knowledge

The knowledge applied for generating HTML pages is similar to the four kinds described in [12] for text report generation. The following techniques are applied for the representation:

1. Conceptualizing Web Page Components and the Document Generation Expertise

The knowledge of source information, either from the data in a database or the facts in other format, can be organized by relating each knowledge element to conceptual semantic network (CSN). Let us take DCMS as the example. As we know from previous illustration of DCMS, the information of a cluster of HTML pages in SMR, including document-inherent information as well as linkage information, is matched to DCB. Therefore, DCB is in fact an abstraction of SMR. The data in DCB are further tied to CSN to make data meaningful. The records in SPC, TOC, FRC, and EUC become elements categorized into different sets, each of which represents a distinctive concept. The records in SPCL, TOCL, and FRCL are elements defined as different relations.

For better understanding of CSN, let us examine a sentence within HTML document *d-0*:

AMAT *is* ***down slightly*** ***today***.

A link is set at the position of ***AMAT***[3] to HTML document *d-1*, and the other at ***down slightly*** to document *d-2*. The third sentence component has its semantic meaning but without any hyperlink. Thus, we have the semantic connections among the three HTML files as shown in Figure 5:

[3]*AMAT* is the stock symbol for Applied Materials.

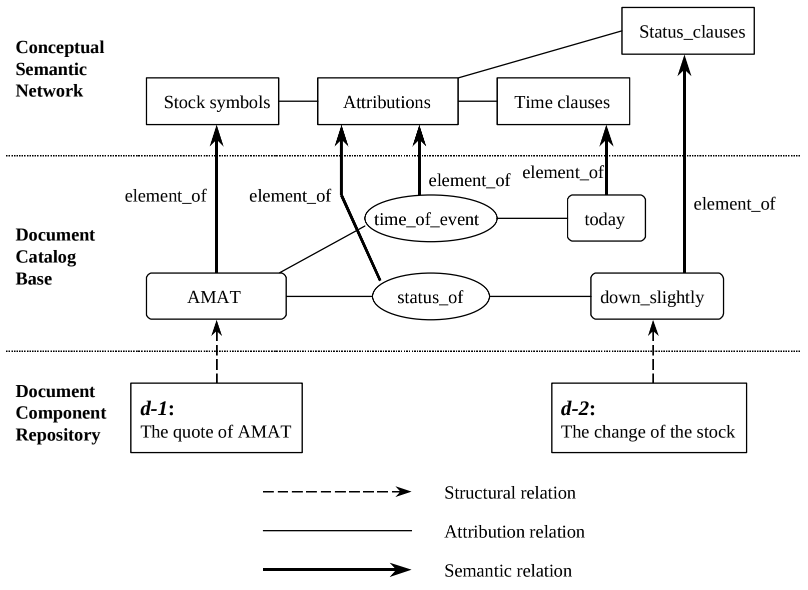


Figure 5 An example illustrating semantic relations among HTML document components.

down slightly is asserted when the change rate of ***AMAT***'s price meets certain condition, say, between 0.1 and 1%; or this could be ***down sharply*** if the change rate of the price falls in the range 2~4%.

An HTML file usually has more than one sentence. Therefore, the structure of every HTML document is extracted as a *Backus Normal Form* (BNF) and placed into one of the fields in its affiliated record in DCB. Similarly, the information in SIR for updating HTML pages is also linked to CSN in the same way HTML pages are treated.

KHDG handles the links among HTML documents, the sections linked in the same document, and related files in other types as semantic relations. The regions where pointers are set are branches of a node identified by the name of the host document in the semantic network. Therefore, Web page production by KHDG is in fact the process of expanding the semantic network with the relationship set by the HTML links.

The concepts used in the rules share the same nodes with other types of knowledge bases. In this way, the rules are able to know who's who for the entities in applied facts or files.

2. Using a Rule-Based Approach to Represent Expertise for Domain-Dependent HTML-Based Publication

Two kinds of knowledge, *domain knowledge for information extraction and analysis* and *HTML composition knowledge*, reside in AAE as rules. The concepts referred by these rules are also linked to CSN, so that the fact knowledge and rules will be consistent and easy to be dealt with.

3. Manipulating Stylistic Knowledge in Templates

Stylistic knowledge is maintained in HDF as templates that can be expressed in BNFs. There are two subsets of stylistic knowledge, that is, structural knowledge and formatting knowledge. Structural knowledge is the knowledge about the structure of contents in a cluster of HTML documents, that is, what kind of information will be embedded for the theme these documents are to support. The following is a set of BNFs for a sample template:

```
<subject> ::=<topic>|<topic>[<subject>]

<topic>::=<opening section><body section><closing section>|<body section>|

      <opening section><body section>|<body section><closing section>

<section>::=<opening section>|<body section>|<closing section>

<section>::=<paragraph>[<paragraph>...]

<paragraph>::=<functional region>[<functional region>...]

<functional region>::=<sentence>|<picture>|<table>|<filler>
```

The second subset of templates is for Web page typesetting. HTML publications updated periodically and the layouts for instant responses to various queries are documented in certain structures. There are also several frequently used typesetting styles for the appearance of the formatted version. This knowledge serves to present HTML documents in a WYSIWYG way. It is somewhat art-related. The constructive components for a typesetting template are the position, size, style, and other such things for elementary components in formulas for certain kinds of contents.

IV. Implementing KHDG

A. Obtaining Up-to-Date Publishing Components from Source Materials

Routinely updated information in SMR is the source for refreshing publications. DAN looks for the source materials in two stages. The goal of the first stage is to extract useful information from SIR. The strategies applied for this are as follows:

- *Scanning the source information in an in-depth first approach guided by heuristic rules* — The source information could be in a large volume, whereas the size of needed output is restricted. Thus the templates in HDF are applied as goals to circumscribe the search on SMR.
- *Utilizing metaknowledge to make reasoning more effective* — There are several ways to apply metarules, such as by rule selection, search strategy decision, etc. [13].
- *Inducing assertions by summarizing extraction* — The induction is implemented in two ways:
  - reaching subgoals by analyzing the data and facts in SIR with analysis rules; the results are assertions in the form of first-order logic;
  - presenting converted forms by executable procedures; the results may be in the form of graphs, calculated values, or other transformed formats.

The reasoning processes apply backward chaining approaches with goal-driven mechanisms.

The results turned out by DAN at this stage are blocks of text that are “concentrated” information following the same syntax as that of templates in HDF, but to be formatted and reorganized for publishing afterward. The approach used for generating natural language texts from data is based on the same principles in [12, 13].

B. Planning Hyperlinks

Composing Web publications at the structural level is actually a hyperlink planning based on the set of components available. When newly extracted composing materials are ready for HTML page construction, DOP is activated by KDHG to carry out the task. DOP works for the HTML publication on the three sources of materials: new composing components, previous versions of publication, and a collective set of constructive materials from DCMS. The decisions to be made are:

- what the hierarchical structure of the HTML pages is;
- what content must be placed in each page;
- where to set up links from a page to the target HTML files or other regions within the same page;
- which document component is to be linked to a specific region in an HTML file, where the topic is consistent with the contents in that linked component;
- how to reuse the document components that are currently used by the old version of the publication.

The planning task could be made very simple by retiring old components in the cluster of templates being used and rearranging new ones to correspondent places. However, elaborately designed publications request more than that. A quality job is evaluated by criteria such as fresh appearance, evolving style, ever-extended information scope, and advanced reader-oriented facilities. The level of DOP will be kept up to a level between these two extremes. To plan HTML links, some rules of thumb are applied by KHDG. The strategies at this phase are:

- using the selected templates in HDF to plan document application structures;
- using the authoring expertise to organize the extracted source materials.

The process of document planning is focused on a semantic structure design for HTML documents. It is to construct a new framework using both the original one and templates in HDF. The structure of this framework is publication material dominated, that is, its structure must fit the information to be housed.

C. Customizing Web Pages

Customizing HTML pages is a process that is for tagging previously produced materials, new components for publication, and planned hyperlinks into WYSIWYG HTML documents under typesetting rules. The following factors contribute much to the quality of publications:

- *the fonts* — This issue includes the decision for the type, size, style, color and shape of the fonts to be used;
- *the position of a document component on the page displayed* — For example, the decision whether to use an in-line image or not;
- *the type of list to be used*;
- *the fancy features used for the pages* — Form, frame, annotation, menu, and user-defined functions are possible choices.

The principles for combining these factors stem from aesthetics, and result in a set of rules for typesetting.

HFG generates an HTML page by consulting the rules in the AAE knowledge base and by referring to templates in HDF. Tags of the HTML language are mainly divided into three types: the basic constructive type for a regular HTML document, the tags for connecting different HTML components or different regions within the same file, and the tags or subtags that make pages fancy. It takes three steps for HFG to finish the task by using these types of tags:

- converting unformatted newly created components into HTML files,
- setting up links among these files plus those inherited from the previous version of the publication, and
- fine-tuning the format of each HTML document according to rules in AAE.

After finishing the task, HFG will rearrange the created HTML cluster into DCMS following indexing rules.

D. Programming Considerations

Web technology has been experiencing rapid change. There are more and more diversified choices. The Windows NT based Web server provides an alternative development environment to that provided by UNIX. Figure 6 shows the architecture for KHDG development environment on the Windows NT platform.

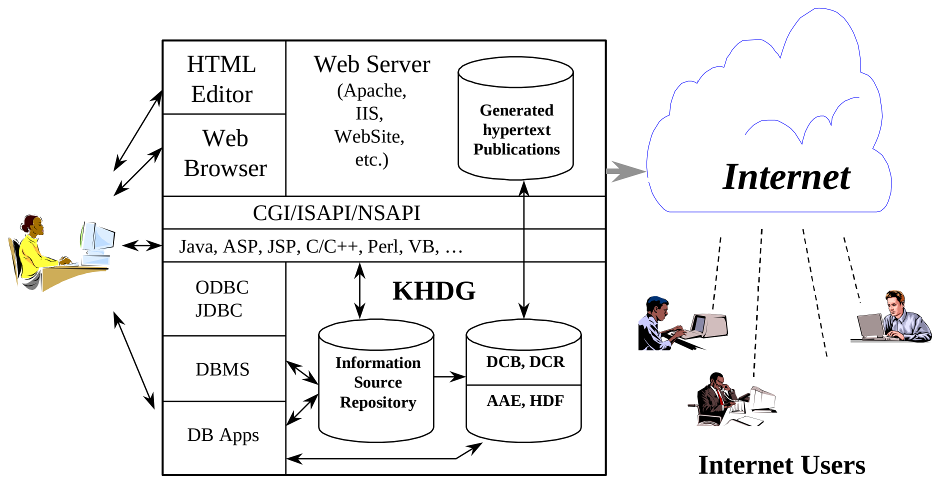

HTML Editor
Web Browser
Web Server (Apache, IIS, WebSite, etc.)
Generated hypertext Publications
Internet
CGI/ISAPI/NSAPI
Java, ASP, JSP, C/C++, Perl, VB, …
ODBC JDBC
KHDG
DBMS
Information Source Repository
DCB, DCR
AAE, HDF
DB Apps
Internet Users

Figure 6. Current KHDG development environment

Basically, three aspects within the development platform underpinning user interface are considered.

1. Web Server Software for KHDG

Generally, we may apply any non-Web system to generate HTML documents. The produced HTML publications can then be moved into a Web server for service. However, this scheme could not cope with online tasks, which request real-time response. To make KHDG more available on the Internet, it is better to use the client–server computing environment provided by Web technology. There are four leading kinds of Web server software on the Windows NT platform available at present: Microsoft's Internet Information Server (IIS) (http://www.microsoft.com/iis/), Netscape's Enterprise Server (http://home.netscape.com/download/index.html?cp=hmp01sser), Apache HTTP server (http://apache.org), and O'Reilly's WebSite Professional (http://www.ora.com). All four of these[4] provide versatile API to most commonly used programming language and database. Take IIS as an example. IIS is tightly integrated with the Microsoft Windows NT server operating system and is designed to deliver a wide range of Internet and intranet server capabilities. It is a native implementation of the current Internet standards for Web servers. By optimizing around

[4]The information on the comparison among them can be found at http://www.tedhaynes.com/haynes1/infoserv/haynes1.htm and http://www.netcraft.com/survey/.

the Windows NT Server platform, Internet Information Server is able to deliver high performance, excellent security, and ease of management. When Microsoft Internet Information Server version 1.0 was released in February 1996, it quickly established itself as one of the fastest Web servers available. This was consistently proven by Web benchmark tests run by companies such as Shiloh Consulting and Haynes & Company, as well as by respected trade publications such as *PC Week* and *Windows Magazine*. Internet Information Server version 3.0 is even faster, improving Web server performance almost 40% over version 1.0. IIS 3.0 proves that the combination of Windows NT Server and the Internet Information Server is a good Web solution for both corporate intranets and the busiest sites on the Internet. The Internet Information Server proprietary API (ISAPI) is roughly five times as fast as CGI. IIS is a complete Web server, including a full set of tools for managing the server and for creating and managing content, and a search engine for finding documents. Its high performance and open extensibility make using IIS a convenient way to build business applications on the Web. At this writing, the latest version of IIS is 4.0.

The NT version of Enterprise Server and WebSite Professional also provide capacity similar to IIS'. Apache is the most widely used Web server in the world, mainly on the Unix platform. It now also supports the Windows NT. Although the Windows version of Apache may not perform as well as that on the Unix systems for which Apache was originally written, the ability to run on Windows gives Apache the ability to run on a large number of Web servers it was not previously able to.

2. Web Scripting

Now the Internet application programming interface is replacing the traditional common gateway interface (CGI) for delivering much faster and more integrated Web-based application services. The main standards available for the Windows NT server are Internet Server API (ISAPI) by Microsoft and Netscape NetServer API (NSAPI) by Netscape. ISAPI enables software developers to create professional applications in C/C++ that outperform regular CGI scripts by approximately 10 times. ISAPI programs compile into DLLs, a Windows standard for dynamically loaded code sections of an executable program. Because of the use of DLLs, the ISAPI program is loaded into the same memory address space as Internet Information Server. Such a loading makes it run more securely, robustly, and efficiently. Many ISAPI programs extend database connectivity and provide filters for incoming and outgoing HTML information. In the same way ISAPI does, NSAPI also greatly improves the performance of Web applications.

Several scripting languages are suitable for KHDG development accounting for the preceding APIs. Born in 1995, SUN's Java (http://sun.java.com/) is fast becoming a dominant programming language for Web-based applications. Java is based on C++ and designed for secure two-way, real-time interactions. When integrated into Web pages, Java applets allow expert graphics rendering, real-time interaction with users, live information updating, and instant interaction with servers over the network. Netscape Navigator and Communicator support Java on Windows NT/95 and most of Unix versions. The Java programming language and environment is designed to solve a number of problems in modern programming practice. The object-oriented feature of Java makes it feasible to set up a practical programming library to be reused. JDBC[5] provides a convenient yet efficient way to access heterogeneous databases with Java programs, where JDBC is a Java API for executing Structured Query Language (SQL) statements.

Perl is another ideal language for Web scripting. Perl version 5.x has been enhanced to have several useful features. For example, Perl 5.x has object-oriented programming capability with well-defined classes and structures; C/C++ programs can be embedded into Perl, which makes Perl more powerful; and the Perl 5 module Win32::ODBC is available for interfacing ODBC[6] on Windows NT. Perl has two important features that make it a good programming language specialized for KHDG:

(i) Perl is an interpreted language, so the new script generated by Perl can be executed directly in the way LISP does; in this sense Perl is good enough for knowledge-based programming, though it is not considered a logic programming language as Prolog is.

(ii) Perl has a powerful regular expression handling function that is convenient for knowledge processing, which is usually in string format.

Besides Perl, Visual C/C++ and Visual Basic are also good Web scripting candidates.

[5] Java Database Connectivity, see http://java.sun.com/products/jdbc/

[6] Open Database Connectivity, see http://www.microsoft.com/data/odbc/default.htm

3. Database Extension

Owing to ODBC and JDBC there is no problem interfacing most of relational database management systems (RDBMS) on Windows NT to Java, Perl, or some other kind of scripting language. ODBC is based on SQL and provides a common interface for accessing heterogeneous SQL databases. This interface provides maximum interoperability: a single application can access different SQL database management systems through a common set of codes. This enables a developer to build and distribute a client–server application without targeting a specific DBMS. Database drivers are then added to link the application to the user's choice of DBMS. JDBC is the most significant new database middleware standard. JDBC defines a call-level interface (CLI) for Java development. It is really a set of Java classes for specific databases and borrows heavily from the ODBC architecture. JDBC provides a standard API for tool and database developers and makes it possible to write database applications using a pure Java API. Using JDBC, it is easy to send SQL statements to virtually any relational database. A combination of Java and JDBC allows a programmer write it once and run it anywhere.

The RDBMS products complying with ODBC–JDBC standards form a long list: Oracle, Sybase, Access, Informix, FoxPro, etc. The local source information[7] and document repository in KHDG can be well maintained in a RDBMS. To extend a database to a knowledge base, a middleware is required to bridge ODBC–JDBC and KHDG kernels. This is in fact an implementation of CSN. Therefore, CSN can be conceived in Perl or Java as is adopted by KHDG for kernel part programming.

V. A Prototype: Smart Stock Information Analyzer

A pilot system named *Smart Stock Information Agent* (SSIA) is an attempt to test the ideas of KHDG. The objective of this system is to provide users the latest analytical information about a stock for investment advice. As we can see knowledge-based methodology has been used quite extensively in the financial field, particularly in the stock market [14, 15]. Also Web-based

---

[7]Here we use *local source information* to distinguish it from *remote source information*. The latter is another type of primary information retrievable from remote web server and is to be processed by DAN jointly with local source information.

online investment services have emerged as a major e-commerce business. For example, E*Trade (http://www.etrade.com), DLJDirect (http://www.dljdirect.com), and Datek (http://www.datek.com) are among the most popular electronic stock brokers in the United States. All these services provide dynamic stock information, including quote, trading volume, historical data, charts, news, research results, reports, major composites, etc., from major U.S. stock exchanges and stock market researchers. Investors read the stock information they are interested in and make a decision after careful analysis. The investors have been trying to obtain as much information as possible by consulting several other relevant stock information sites before they throw money into the fluctuating stock market. Other information sources include Yahoo's financial information server (http://quote.yahoo.com), stock forums (e.g., http://www3.techstocks.com/~wsapi/investor/stocktalk), initial public offering (IPO) information (e.g. http://www.techweb.com/cgi-bin/IPOData/reports/hitech97.html), a broker's homepage, and the stock issuing company's homepage. Although there are abundant resources available, searching around and sifting for most crucial clues are time-consuming. Hence, we define SSIA's missions as follows:

Analyze a set of the latest stock information and provide an evaluation of the stock;

Find relevant reference information from available sites;

Generate a customized HTML page for the user.

Due to the limited time budget and resources, SSIA is written in Perl 5 and runs on Apache under Unix to make use of scripting resources did before. Treatment of several implementation issues specific to SSIA are given in the following sections.

A. Making Remote Web Servers the Information Sources

SSIA obtains stock information, such as quotes, news, and reports, from remote databases that are accessible from web sites. Each of this kind of site has a record in SPC, the top level of the document catalog base. The record contains all needed messages of the site, including the URL, the created date, the creator, the interested data item names, data extraction syntax (or extracting subroutine path), and other attributions. SSIA launches an HTTP client process in the background to access remote web servers whenever there is a need to retrieve data from the database in that server. Currently, SSIA obtains quotes from Yahoo's service. SSIA has to extract acquired data from the HTML-formatted responses. The subroutine for the job is either in an existing library or dynamically generated according to the extraction syntax housed in the site record in SPC.

B. Representing AAE in Rules

SSIA maintains AAE in the following syntax:

```
<rule> ::= <rule identifier><antecedent> -> <consequence>

<rule identifier> ::= Number

<antecedent> ::= <condition>[ OR|AND <antecedent>] | <rule status>

<condition> ::= Regular logic expression

<consequence> ::= <assertion>|<action>

<assertion> ::= A sentence for expressing analytical results

<action> ::= An executable function
```

The following are examples of the analyzing rules:

```
1|$stock_change_per < 0.1 && $stock_change_per > -2

       -> "$stock_name is down $stock_change"

12|$stock_price >= $year_hi

       -> "The stock price is at the highest level in 52 weeks" 14|$assert10 &&
($assert4 || $assert1 || $assert2)

       -> "Suggestion: $symbol is strong buy"

22|! $stock_price && $symbol -> &get_quote($symbol)
```

Using Perl's string processing functions and its `eval` function, the preceding rules can be handled easily within the Perl program.

C. Maintaining HTML Documentation Framework in a Hierarchical Structure

SSIA handles an HTML document at top level in three parts: header, body, and bottom. Each part then has several templates to be chosen by heuristic rules. For example, a body template is

```
<center>

<table>

<tr>
```

```
        <td>$block_comm<p>$block_curve<p>$block_trade</td>
        <td>$block_quote<p>$block_dj</td>
        </tr>
        </table>
        </center>
```

Components in a template can be generated from the templates in a lower level, whereas these templates may refer to even lower levels of templates. Therefore, these templates form a top–down tree structure. Each node on the tree is related to a template file. The construction procedure for an HTML page is actually the pruning of the tree according to the structure of information ready to be published. Collecting information for publishing is goal-driven, guided by the knowledge obtained from the previous page design. In other words, this is based on a previously generated HTML page structure tree. This goal-oriented information searching and sifting can reduce the cost of arbitrary search. However, the search is not necessarily to repeat previous paths, but could diversify according to contemporary search results. Therefore, SSIA works out a new page structure with the final search results.

SSIA allows a user to input a stock symbol and returns search results in a simple page. The information on the page includes basic information about the stock, a few U.S. stock market indexes, a short analytical comment, and some useful links.

VI. Summary

Although the rapidly expanding World Wide Web is a promising basis for online publishing on the Internet, it also presents many unsolved problems for developers and researchers. In this chapter we propose a scheme to meet a specialized authoring and maintenance need for Web document production. The idea is to apply knowledge-based methodology in an automated HTML document generator, named KHDG. Utilizing developments on the Internet in 1990s, KHDG adopts fresh ingredients to tackle the new problems related to Web publishing, which go beyond the previous plain text generation, such as formatting and links planning. In fact, KHDG is a by-product of the CODE project, which is aimed at worldwide collaborative applications for educational purposes. Other related projects, such as NCEC (which stands for network-based continuing education for China) proposed in 1995, raises the same demands on improving the productivity for creating and maintaining a set of Web-based materials. KHDG and CODE/NCEC will share the outcomes to reinforce each other. For example, the idea of DCMS has been a part of CODE's structure, whereas components developed by CODE, such as *Localizer* and *Indexor*, in turn are available for KHDG. Since proposing the idea of KHDG, some practical progress in similar projects has been achieved.[8] SSIA, a pilot application in stock information analysis, is aimed in the direction of electronic commerce. It has changed the previous source information retrieval path from local databases to remote databases that are interfaced to the Internet by Web technology. As e-commerce becomes popular in the commercial world, we are expecting that there will be stronger demand for this application. However, improving the productivity of HTML publishing is still far from satisfactory. One of the major reasons is that the fast moving technological frontier has been continuously posing new challenges to the research. Hopefully, KHDG can absorb most advanced techniques to keep pace with the development of Internet technology. For example, Windows NT can support several kinds of Web server software with full Internet services, which was at a very initial state when KHDG ideas were formed in 1995 (Lin et al. 1995). To implement DCMS is not a problem, but we need to work further to centralize the widely distributed HTML into an integrated knowledge base. Refining the model of HTML document link planning is always a challenge to KHDG because HTML has been constantly evolving. This requires KHDG to be able to make use of the features provided by the latest version of HTML.

References

[1] Kalakota, R., and Whinston, A. B. *The Frontier of Electronic Commerce.* Addison–Wesley Longman, 1996.

[2] Kalakota, R., and Whinston, A. B. *Electronic Commerce: A Manager's Guide.* Addison–Wesley Longman, 1997.

[8]For example, according to a press release from Cognition Technologies Corp (US), the company launched software, DocuSMART, that can automatically generate hyperlinked HTML pages with a rule-based approach (CINET-L Newsletter, No. 50, September 1, 1995, http://www.cnd.org/CHN-China/CND-China.95-09-01.html).

[3] Anand, T., and Kahn, G. Focusing knowledge-based techniques on market analysis. *IEEE Expert*, 8(4): 19–24, 1993.

[4] Mertons, P. Derivation of verbal expertise from accounting data. In *Expert Systems in Economics, Banking & Management*, (L. F. Pau, J. Motiwalla, Y. H. Pao, and H. H. Teh, Eds.), pp.341–350, North-Holland, Amsterdam, 1989.

[5] Lin, Z. RASF—Automating routine analysis of financial data. In *Expert Systems in Economics, Banking & Management*, (L. F. Pau, J. Motiwalla, Y. H. Pao, and H. H. Teh, Eds.), pp.69–75, North-Holland, Amsterdam, 1989.

[6] Lin, Z., Huang, Y., Huang, L., and Wang, T. Automated verbal report generation based on economic databases — cases and technological issues. *AIEM3*, Portland, August, 1993 (a full paper).

[7] O'Leary, D. E. The Internet, intranet, and the AI renaissance. *Computer*, 71–78, January 1997.

[8] Lin, Z., Hämäläinen, M., and Whinston, A. B. Intelligent HTML document generation for routine publication based on the Internet. PACES'95, May 1995 (a full paper).

[9] Lin, Z., Hämäläinen, M., and Whinston, A. B. Knowledge-based HTML document generation for automating web publishing. *Exp. Sys. Appl.* 10:381–392, 1996.

[10] Shneiderman, B. Designing information-abundant web sites: issues and recommendations. *Int. J. of Human–Comput. Stud.*, 47(1):5–30, 1997.

[11] Higa, K., Morrison, M., J., and Sheng, O. R. L. An object-oriented methodology for knowledge base/database coupling. *Commun. ACM*, 6(35):99–113, 1992.

[12] Lin, Z., Huang, Y., Liu, Q., and Wang, T. The knowledge structure of verbal reports and its representation. In *The Proceedings of the International Conference on Artificial Intelligence in Industry and Government*, (E. Balagurusamy, Ed.), Nov. 23–25, 1989, Hyderabad, India, pp.600–611.

[13] Wang, D., Lin, Z., Wang, T., and Huang Y. Application of meta-knowledge to analyzing and reasoning system on macroeconomic data. *Information and Systems*, (S. Hu, and S. Jiang, Eds.), Vol. 2, pp.840–843, 1991.

[14] Liu, N. K., and Lee, K. K. An intelligent business advisor system for stock investment. *Exp. Sys.*, 14(3):129–139, 1997.

[15] Zopounidis, C., Doumpos, M., and Matsatsinis, N. F. On the use of knowledge-based decision support systems in financial management: A survey. *Decision Support Sys.*, 20:259–277, 1997.