A SPECTRUM OF AUTOMATIC HYPERTEXT CONSTRUCTIONS

RICHARD FURUTA, CATHERINE PLAISANT and BEN SHNEIDERMAN

Human Computer Interaction Laboratory and Department of Computer Science, University of Maryland, College Park, MD 20742

We describe our experiences with four separate conversions from paper documents into hypertext and discuss the lessons we have learned. The paper document’s organisation affects the ease with which it can be converted and the appropriateness of the resulting hypertext. The form of the paper document’s machine-readable ‘markup’ description affects the ability to transform the structure automatically. Designing the link structures that tie together the parts of the hypertext takes special care in automating, as badly-designed and incorrectly-formed links destroy the integrity of the hypertext. Overall, each of the conversions followed the same basic methodology, providing the handle for the development of ‘power tools’ that can be applied to simplify subsequent conversions.

1. INTRODUCTION

In a very few years, hypertext systems have become immensely popular with computer users. Hypertext databases can be very helpful in locating information in a large body of inter-connected articles.

Of course, not all applications benefit from hypertext organisation and not all versions of a hypertext document are equally easy to traverse. There is a growing body of experience and scientific evaluation on how and when hypertext succeeds.

As the enthusiasm for hypertext increases there is a great desire to convert existing documents into hypertext format. Some documents must be restructured and rewritten to take advantage of the hypertext environment. However, if the underlying structure of a document contains many relatively short pieces that cross-reference each other then there is a strong possibility that automatic importation of the document with automatic construction of the links is possible.

A stereotypical example is a university course catalogue that contains titles and descriptions for each course, plus references to prerequisite, co-requisite, related, and follow-on courses. In addition, each course listing might mention professors who teach it, textbooks used, or departments requiring the course. All of these can be links to other parts of the catalogue. If the file containing the catalogue has an explicit structure to it, then it should be possible to automatically import and create links. This is often done by writing a special-purpose program to convert from the existing format to the input format of the hypertext system. However, as we gain an increasing amount of experience with such special-purpose conversions, it is becoming apparent that many conversions follow the same pattern; a pattern that can be represented by a general framework.
In the framework that we have come to use, we first identify the *logical components* of the input that correspond to components in the hypertext. We describe the form of the logical components with a template or simple grammar, extract the components with a parsing program, and finally automatically create the hypertext database, rearranging the ordering of the logical components when appropriate.

The Hyperties hypertext system was developed in our lab, and our experience with it dates back to 1983. The IBM-PC version of the Hyperties browsing and authoring tools are now available commercially, and research in our lab continues, based on a Sun version.

In this paper we describe our experiences, good and bad, with four projects that built hypertext databases from existing paper documents. In section 3, we describe the conversion of eight scientific papers into hypertext for the Hyperties version of ‘Hypertext on Hypertext’. Scientific papers do not fall directly into the category of documents that are regularly-structured and richly cross-linked. Consequently, the papers required a significant amount of restructuring before conversion into hypertext — our experience suggests that this restructuring would be difficult to automate fully.

In the second project, described in section 4, we converted a more regularly-structured course catalogue into hypertext. Here, we achieved a fully-automatic conversion, but the recognition of the structures in the input was made more difficult because the input file did not contain much description of either the document’s logical structure or its physical appearance. Our initial experiences with automatically generating links are also reported here.

The third and fourth projects also achieved fully-automatic conversion, converting a technical report abstract listing (section 5) from markup describing the physical appearance, and converting a dissertation abstract list (section 6) from a logically-based markup.

In the final section of the paper, we reflect on the lessons we have learned from these four projects and from approximately two dozen other projects in which we built databases manually. First, even with the diversity of sources and target databases, it is possible to provide tools that can help automate the potentially tedious task of loading a database and making the links. However, the characteristics of the source and the destination affect the degree to which the transformation from source to target can be carried out without manual assistance and the degree to which the specification of the transformation can be reused in subsequent transformations.

We have observed that the transformation can become quite automated for some source—target pairs and describe some of the identifying characteristics of the source and of the target. For those situations in which a fully-automatic transformation is achievable, we have found that steps
taken in carrying out the independent transformations can often be described by the same methodology.

A word of caution is appropriate as well. Not all sources make effective hypertext databases. The designer must be cautious in identifying sources that are appropriate for conversion to hypertext. In general, the 'golden rules of hypertext' should be considered when deciding if a conversion is warranted:

- there is a large body of information organised into numerous fragments,
- the fragments relate to each other, and
- the user needs only a small fraction at any time.

Creation of a hypertext involves two separate but interrelated activities: (1) generation of the individual articles in the hypertext's database, each with their own internal structure and (2) identification of the interrelationships among the separate articles (forming the contextual links). In both cases, automation becomes especially attractive when the source document follows the 'golden rules', with regularly structured fragments whose boundaries may be detected unambiguously.

**Related work**

Achieving transformation of documents from paper to hypertext form has significant application in the reuse of previously-developed documents. Egan, et al., consider one approach to the reuse of pre-existing documents in SuperBook which provides additional means for traversing a paper document's interactive display. Nunn, et al., have performed a conversion from the IBM REXX manual to Hyperties. The conversion was manually-specified. Based on that experience, they designed and implemented a semi-automatic conversion for a portion of the manual. Glushko reports on the conversion of an encyclopaedia into hypertext, emphasising that a straightforward translation of a document into a hypertext will not result in a good product in the absence of careful attention to the design of the hypertext. Texas Instruments has released a tool called HyperTRANS that permits iterative specification of the transformation from text into hypertext.

As will be seen in the discussion of our case studies, a central consideration in the development of an automatic conversion is the extraction of the original document's structure. General recognition of structure in the absence of markup remains a research problem in the area of document preparation systems (see, for example). Recognition of structure with assistance from markup is often more manageable. For example, Mamrak, et al., consider the problem of recognising and converting structure in markup languages with Chameleon. Such techniques will be of application within our domain as well. However, Chameleon's extraction of structure requires manual assistance. The more regular (and limited) structure of our problem domain has helped us in developing fully-automatic translations.
2. THE FORMAT OF THE HYPERTIES DATABASE

The overall model of a Hyperties database is based on the metaphor of an 'electronic encyclopaedia'. The database consists of a set of short articles, related to one another by links, which are displayed as user-selectable highlighted strings within the body of the article. Each article is further subdivided into three fields: the title, a short string naming the article; the definition, a short phrase describing the article; and the body of the article. The display of an article shows both the title and the body. When a user selects a link, the destination article’s title and short description are shown in a separate window. Confirming the selection causes the source article’s display to be replaced by the destination article.

The full PC-based Hyperties system permits textual, video, and graphical article bodies, and has many other features. It is not necessary to consider these additional characteristics in the context of this paper’s topic. It is worth noting, however, that application of these other features, particularly the enabling of fast string search over the collected bodies of the articles, greatly enhances the usability of converted databases.

3. CONVERSION 1: HYPERTEXT ON HYPERTEXT

The July 1988 Communications of the ACM (CACM) included a special section containing papers on Hypertext research — papers drawn from the pioneering Hypertext conference held in North Carolina in 1987. The presentation of the articles in the special section was not only in paper form — three separate versions of the articles were independently-prepared for the Hyperties, HyperCard, and KMS systems. In this section, we describe the preparation of the Hyperties version, directed by Ben Shneiderman. The conversion into Hyperties was manual, relying only on the Hyperties Authoring System’s import function, which permits importation of content one article at a time.

The special section of the CACM contained eight papers which were cut into 86 Hyperties articles, 38 figures, and 120 references (reference handling is described below). The copy editing was completed at ACM Headquarters by early June 1989. The first draft of the disk was to be delivered by late June for review by editors and authors. The preparation of the Hyperties version required solving both practical and also conceptual issues. A practical challenge, requiring reworking of some of the figures, was how to represent the articles’ figures in sufficient detail, given the 350 x 200 pixel resolution of the targeted IBM-PC CGA screen. The size of the Hyperties display window (17 lines), the need to represent mathematical equations, and the need to represent special characters also required solutions.

The major challenge in converting the eight papers to hypertext form was not in these technical aspects but in design of the logical structure. The original papers were independently written for journal publication — in
linear form with minimal cross referencing across papers. Simply putting
the text on the computer screen would degrade the contents because typical
computer screens are approximately 30% slower to read than typewritten
paper documents, more clumsy for page turning, and annoyingly small.

Some of the articles already contained explicit natural links, for example
the Guest Editor’s Introduction. Shneiderman added his own overview that
offered a set of links to topics that were covered in several papers, because
the papers didn’t provide the links, and extracted some commentaries from
Nielsen’s Hypertext 87 Trip Report since it provided an alternate viewpoint.
Links were made between the author names and their biographies.
Additionally, each bibliographic reference within an article was linked to
an article that contained the associated citation. The final database had 307
articles occupying 360 Kbytes.

The hard work of cleaning up the formatting and putting the papers in
Hyperties required part-time effort over six weeks by Yoram Kochary and
Andrew Sears. While some portions of the conversion could certainly be
automated, it seems likely that, even with automation, another conversion
of a similar collection of eight scientific papers would require a similar
amount of overall effort.

4. CONVERSION 2: NCR MANAGEMENT COLLEGE COURSE CATALOG

In this section, we describe the conversion into Hyperties of the NCR
Management College Course Catalog, which includes abstracts for all
courses offered under the aegis of the NCR Management College. The
conversion was directed by Catherine Plaisant. (Figure 1 shows a sample
of the resulting Hyperties database on a Sun-based experimental version of
Hyperties).

The NCR catalogue is an attractive candidate for fully-automatic
conversion. The catalogue information is naturally divided into a number
of discrete, relatively small units (i.e., separate course descriptions). Each
course description has a fairly consistent structure (e.g., each course
contained name, number, length, tuition, intended participant, description,
prerequisites, content, instructional method, enrollment procedures, etc.).

Each course description was converted into a separate Hyperties article
(the 400 Kbytes of source produced 193 articles, and 30 additional articles
containing overviews, maps, etc., were added manually). The title of the
course was used as the article title. The course number became a synonym
for the title. No definition could be isolated in the article: some courses have
a short description but most of the time the text of the description is too long
to be used as a Hyperties definition. Instead of truncating the course
description we decided not to provide any definitions. The article’s content
is the remaining text of the course description, and references to prerequisite
courses in the content are linked to the appropriate description.
FIG. 1: Sun Hyperties article display

184
AUTOMATIC HYPERTEXT CONSTRUCTIONS

Specification of the conversion was not as simple as might be hoped because only minimal markup was available in the input source. Figure 2 shows an extract taken from the input. The beginning of each course description is marked by a line containing an ‘*’, but the fields within the description are distinguished only by context. Because neither structural nor physical formatting commands are available, the recognition of the subfields (title, course number, etc.) had to be based on pattern recognition done on the context of the text. While the source file seems very regular when scanned visually, numerous deviations from the regular form complicate the conversion. For example, the course description’s title is found in a number of different formats:

1. TITLE in uppercase on the first line of the article
2. subtitle (the module name, in lowercase)

   blank line

   TITLE (in uppercase)

3. TITLE: subtitle (TITLE in uppercase, subtitle in lowercase)

Only form 1 was identified during the initial visual scanning of the source file. Cases 2 and 3 were only discovered during the first iteration of the conversion process: they correspond to courses that were modules of a multi-module course. Once all three forms were identified, only exceptional cases remained. For example, the rule expecting a title in uppercase had to be made more flexible to accept titles like ‘LEARNING dBASE 2’, which contained both upper and lowercase letters in the title.

It is possible to put a lot of hope in the development of pattern matching techniques that will simplify this task. Nevertheless, in this case, our experience shows that the conversion had to be closely monitored to verify the appropriateness of the conversion. The source data was too large to identify all the special cases manually but fortunately limited enough to allow the monitoring of the conversion.

With such a database the possibility of extracting a piece of the source file to be treated separately becomes a major desirable feature of the conversion program. The ‘faulty’ piece of source can then be edited and reinserted in the source (e.g., to correct a spelling error that bugs the conversion process), definitively evicted (e.g., to eliminate duplicates) or simply separated from the flock for special treatment. We also believe that it is preferable (when possible) to ‘upgrade’ the source file in order to have it fit the general form known by the conversion program, rather than to fix the output of the conversion.

Adding context to the database

The semi-automatic importation of all the course descriptions in a Hyperties database was satisfactory. However the resulting collection of Hyperties
articles remains a puzzling unusable mass of information asking for additional structure. Therefore we manually added supplementary articles to provide better access to the information (e.g., outline of the catalog) and also improve the general appeal of the database (graphic titlepage, maps of the course locations, etc.).

We also structured the database by recreating the implicit links grouping related articles. For example we created lists of modules belonging to a same course or lists of courses given in French or German. Once again the grouping of articles requires some pattern recognition of the articles’ content. Sometimes a simple string search over the articles is enough, thereby making possible the use of available tools (e.g., grep in UNIX) to create new articles from the result of the search. For more complex cases (like the grouping of modules of a common course) we had to add new functions to the conversion program and rerun it.)
Links are specified by replacing each occurrence of a term found in a glossary with the corresponding link. Each glossary term becomes the title (or a title synonym) for an article. The glossary terms were taken from 2 glossaries: (1) an automatic glossary created during the conversion and listing all the course titles and synonyms, and (2) a manually-created glossary listing the titles of the articles created separately. Some article titles were not included in the glossary — for example Hyperties defines an article titled ‘Introduction’ but we didn’t want every word ‘introduction’ to refer to that article. While the glossary mechanism performed well, some anomalous cases arose:

- Recognizing a reference that is included in another one (e.g., ‘GROUP MANAGEMENT’ and ‘GROUP MANAGEMENT - CAI’): In these cases, the longest matching string was preferred.
- Too common terms: One of the course titles was ‘Working’. This course title was referred to several times in the other articles but this string also provoked numerous irrelevant links (e.g., ‘Attendees will be working on . . .’).
- Self-referencing links: A simple search and replace will create links from an article to itself, which should be eliminated.
- Cluttered links: A commonly used word (such as ‘NCR’ in this database) may result in many separate references within an article. In such cases, it can be more appropriate to identify only the article’s first use of the term as a link.
- Inadequate recognition: Obviously, a simple pattern recognition is not always enough to identify a possible link. There are several meanings to ‘press a suit’.

Because an inappropriate link will undermine the user’s confidence in all of the links, it may be appropriate to provide a different presentation for automatically-created links (e.g., different colour or warning in the description), to let the user know that there is a slight possibility that the link might not be relevant. To eliminate any errors, the conversion tool program could provide a user interface allowing the editor to verify every link (e.g., query search and replace). In such cases, it is important that the conversion process be monitored to assure the quality of the result.

5. CONVERSION 3: UMIACS ABSTRACT LISTING

The University of Maryland Institute for Advanced Computer Studies (UMIACS) publishes an extensive series of technical reports. Frequently, reports are issued jointly with other departments on the University of Maryland campus; most often with the Computer Science Department (CS). Once a year, a listing is distributed of the collected abstracts from the previous year’s technical reports. In this section we report on the fully-automatic conversion of the troff markup for this listing into Hyperties. The conversion was directed by Richard Furuta.
Efficient first and second order adaptive learning algorithms of stochastic gradient descent variety are described. Both algorithms can automatically adjust step sizes to achieve optimum convergence rates. Various theorems concerning the convergence properties of these algorithms are discussed.

FIG. 3: A sample UMIACS abstract

Efficient first and second order adaptive learning algorithms of stochastic gradient descent variety are described. Both algorithms can automatically adjust step sizes to achieve optimum convergence rates. Various theorems concerning the convergence properties of these algorithms are discussed.

FIG. 4: TROFF markup of sample UMIACS abstract
Figure 3 shows the printed form of an abstract from the UMIACS report listing, and figure 4 shows the portion of the troff markup that describes it. Figure 5 shows the display of the Hyperties article created by the conversion (the target of this conversion was the PC-based commercial version of Hyperties). In this article, the assigned technical report numbers have been used as the article's title (the top line of the display), the content of the article is formed from the abstract's title, followed by the abstract's author, followed by the abstract itself. A given abstract may be associated with several campus units; the selected abstract is associated with both UMIACS and also with the Computer Science Department. We generate a list of the abstracts associated with each of the organisational units, as well as keep track of where the current abstract is in the collection of abstracts as a whole. The Hyperties article has been augmented with three lines of links reflecting the position of the present abstract in each of these lists of abstracts. Selecting "next" or "previous" takes one to the following or preceding article in the relevant list. Selecting the highlighted name in the middle takes one to a listing of all the articles in the selected list.

FIG 5: Hyperties article corresponding to sample UMIACS abstract.

We now consider the design of the translator that structurally converts the UMIACS abstract listing. The input source's markup in troff describes the appearance of the printed output, and not the logical structure of the document. Fortunately, the use of troff commands was relatively consistent within the markup, although in a few cases, alternate descriptions were
found producing output that was similar in appearance to the markup used in other portions of the input. We described the expected sequences of input commands through a series of regular expressions. The regular expressions were designed based both on the troff commands found in the input but also on phrases found within the text itself. Alternate markup command sequences were recognized and aliased to the primary sequence. The regular expressions were used to label the arcs of a deterministic finite automaton (DFA). As the input stream was read, it was matched against the regular expressions on arcs leading from the current state. When a regular expression was matched, a specified routine was invoked on the input (the exit routine), the DFA's state was changed, and another routine invoked on the input (the entry routine). The job of the exit routine was frequently to close a data structure in the internal representation associated with the previous state while the purpose of the entry routine was to perform the necessary initializations for a data structure associated with the new state.

**INDEX OF INSTITUTIONS**

University of Kent at Canterbury (United Kingdom)
- RELATIONSHIP-BASED INTERACTIVE GRAPHICAL DIAGRAM EDITOR.
University of Leeds (United Kingdom)
- KNOWLEDGE-BASED SYSTEMS IN TOPIC LEARNING.
University of Maryland College Park
- RELATIONSHIP BETWEEN MODES OF GUIDANCE IN CAI DIALOGUE...
- TRADEOFFS IN VISUAL SEARCH FOR TARGETS ON A CRT DISPLAY:...
University of Minnesota
- COMPARISON OF ACHIEVEMENT AND ATTITUDES OF POSTSECONDARY...
University of Nottingham (United Kingdom)
- NEW MODELS OF NATURAL LANGUAGE FOR CONSULTATIVE COMPUTING.
University of Pittsburgh
- SUCI: SELF-ADAPTIVE USER-COMPUTER INTERFACE.
University of South Dakota
- MENUA: A KNOWLEDGE-BASED MENU DESIGN EXPERT SYSTEM.
- MUSHING AND SCROLLING AS ALTERNATIVE INFORMATION:...
University of Southern California

**FIG 6: A portion of the institution index**

The overall result of parsing the input was an internal tree representation of the entire database. The node corresponding to an abstract consisted of fields containing the generated filename that was to be associated with the corresponding Hyperties article, the list of the abstract's assigned technical report numbers, the abstract's title, its author list, and the body of the abstract.

The internal tree was then threaded, producing one thread that connected the articles sequentially, in order of definition, and one thread for each of
the defined technical report series. (Three different series were defined in this listing: the previously mentioned UMIACS and CS reports, and a smaller sequence associated with the TAME software engineering project.)

The internal threaded tree was traversed, and as abstracts were encountered the corresponding Hyperties article was generated. As each article was generated, links were formed by consulting the threads that ran through the corresponding abstract in the internal threaded tree. A separate index listing of reports for each of the defined report series was generated by traversing the associated thread list in its entirety, collecting and filtering the appropriate subfields as abstracts were encountered along the thread.

102 abstracts were extracted from the 1988 UMIACS report listing. The \texttt{troff}-based input markup file was approximately 125,000 bytes in length. Development of the converter took perhaps three weeks of time, but much of this effort was directed towards creating generally-reusable library routines.

Command Languages  
Direct Manipulation  
Graphics  
Individual differences  
Intelligent interfaces  
Natural Language  
Social issues  
Training & Education  
User interface styles  
Voice  

Decision Support Systems  
Expert Systems  
Guidelines  
Input devices  
Menus  
Online help  
Theories  
UIMS  
User manuals

FIG 7: Keywords in human-computer interaction

6. CONVERSION 4: UNIVERSITY MICROFILMS DISSERTATION ABSTRACTS

In this section, we report on the conversion of a collection of 98 University Microfilms dissertation abstracts. The major goal of the selected dissertations was to refine and evaluate user interfaces for human-computer interaction. They were collected by Ben Shneiderman and date from 1985 through early-1989. The design of the Hyperties database was by Ben Shneiderman and John Kohl. The conversion was fully automatic and was directed by Richard Furuta (with program support provided by Degi Young).

The major intellectual effort in designing the Hyperties database was to select the organizational structure of the indexes so that access would be made simple. We chose to permit access by a set of topics, author names, university names, and department or discipline names, rejecting other indexes such as dates, country, order number, or title. Figure 6 shows a portion of the automatically-generated university name index.
Each abstract was manually augmented with a list of topic keywords. Choosing the topic names presented an interesting challenge since there are no accepted taxonomies of work in human-computer interaction. Our first attempt resulted in a list of 65 terms which proved to be too broad for this project. Several attempts were made at a shorter list until we were satisfied that we had usefully categorized the abstracts with 21 terms (see figure 7 for the final list).

The structure of the translator for this dissertation abstracts database differed from that of the UMIACS conversion of the previous section, reflecting the different characteristics of the two input documents and the different features of the two Hyperties database designs. The input markup file was specified in a custom-designed logical-object-oriented language, with fields representing each of the logical subdivisions of the abstract (e.g., author name, institution, abstract title, abstract body, etc.). Consequently, the input parsing task was trivial to implement, requiring much less than a week of effort. The dissertation abstracts were substantially longer in length than were the UMIACS abstracts (the input markup occupied about 235,000 bytes). For this reason, the Hyperties articles were generated as the input source was parsed, and the content of the input was not retained in an internal data structure. Fields of the input representation were slightly reordered in the output Hyperties article, and some fields were slightly filtered (e.g., the dissertation title field was truncated to 60 characters and used as the Hyperties article's title).

As noted above, the design of the Hyperties database incorporated four index lists. In addition to producing the Hyperties article, the parser also produced an external file presenting this information and the generated filename for the article. The records in this file were passed through four separate filters which produced sorted abstract lists corresponding to the four desired indexes. Each of these lists was then processed to produce a separate Hyperties article containing the sorted index, which was then merged into the database.

7. DISCUSSION AND CONCLUSIONS

The complexity of the specification of an automatic transformation from paper-based document into hypertext is affected by the domain of the document, by the characteristics of the original document, and by the characteristics of the target document.

A lesson that we have learned from the conversions reported here, as well as from other document conversions, is that there is a higher payoff in automatically transforming a document that is regularly and repetitively structured than one that is freely structured. This may be seen in comparing the conversion of the "Hypertext on Hypertext" database, requiring individualized re-organization and editing, with the three other conversions. Although conversion of each of the scientific articles from CACM followed
the same broad pattern, the specific manipulations performed differed for each. Such conversions can be made easier through development of tools that handle the mechanical aspects of the transformation. However, since much of the work in converting the database was specialized for the specific article being converted, it is unlikely that the effort spent in specifying a completely automatic conversion would greatly reduce the effort in converting a new collection of scientific articles.

The remaining three cases were more regular in structure, and here leverage is gained through specification of fully automatic conversion. While our first set of 100 dissertation or technical report abstracts took several days of work to design the program to do the loading, subsequent sets of dissertation or technical report abstracts could probably be loaded in minutes or hours. While additional preparation may be necessary if the source database is inconsistent and additional cleanup may be necessary if there are supplementary articles or links that must be included in the target database, the time spent in specifying the initial automatic transformation results in a savings for subsequent transformations.

We have emerged with a clearer and quite general methodology for doing automatic loading of hypertext databases from existing source documents. We recognize these steps:

1. Identify which components of the source database will be transformed into the nodes or articles of the target database. For example, it was clear that each dissertation and technical report abstract would form an article, but it was much more difficult to recognize which sections of the eight scientific papers in the July 1988 CACM would form appropriate articles. Once the mapping from source to target is clear, then tags can be found or placed in the source database.

2. Decide on what links are appropriate across the articles. Sometimes the links are obvious such as course pre-requisites in the NCR database, other times the implicit links must be made explicit, as in the CACM database. Sometimes there will be links which merely thread through a sequence of articles, such as the links connecting sections of the CACM papers.

3. Decide on what additional articles must be created to provide indexes. For example, in the dissertation abstracts database, articles were automatically created that provided an alphabetic list of authors and abstract titles, of institutions, and of descriptors.

4. Reformat if necessary to match the hypertext delivery medium: indenting, centering, vertical and horizontal placement, cleaning up of widows and orphans, capitalization, fonts, etc.

5. Finally, additional material, the “wrapper,” such as a title page, preface, introduction, summary, conclusion, etc., may be added to complete the project.
When the document is regularly and repetitively structured, a practical consideration is the degree to which the structure is reflected in the markup representation. Application of our structure-conversion techniques have been most successful when the structure is well-defined and explicitly represented in the input document's markup. Partially-specified or ambiguously-specified structure in the input's representation requires a more complicated transformation specification. Ease of link detection is similarly dependent on the completeness of the original document's markup specification.

Even when the markup representation of the document is not available, it may still be easy to specify a conversion if the specification of the input document follows a strict template. (See, for example, the cataloging rules used by libraries [Gor81, GW88].)\textsuperscript{14,15} Cases such as the NCR course catalog, however, are difficult to handle by template as idiosyncratic variations between entries are present.

It is important to remember that the technical aspects of transforming the document are not the most critical of the tasks in successfully producing a usable hypertext database. The most significant task is that of designing the database — a process that is manual. As with structural transformation, the potential for reusing elements of this design is higher for regularly-structured documents than it is for general scientific papers.

Even for transformations that do not benefit from automation, a variety of tools seem possible and helpful. Software tools may be applied to restructuring the source document, to facilitating the loading, and to polishing the target database. As we gain more experience through separate conversions, common design features are becoming apparent. Identification and reuse of these features will help reduce the cost of later designs. By analogy, while there is no single automatic tool for turning a tree into a house, there are many power tools for trimming the tree, assembling the house, and refining the interior.

ACKNOWLEDGEMENT

We appreciate the support of NCR Corporation in providing partial funding for this research.

REFERENCES


