Document Translation: Dissertations and Technical Reports

Kaushal Dalal and Edward Fox

TR 93-31

Department of Computer Science
Virginia Polytechnic Institute and State University
Blacksburg, Virginia 24061

September 21, 1993

Document Translation: Dissertations and Technical Reports

Kaushal R. Dalal
Edward A. Fox

September 20, 1993

1 Introduction

This report describes the work on document translation, electronic publishing and integrating WATERS and Techrep carried out during Summer 93. The project was funded by the Graduate School of Virginia Tech. The documents used for the project were Ph.D. dissertations and in related efforts, technical reports.

Large numbers of documents in the form of reports, articles, journal papers, dissertations, etc. leads to the problem of storing and accessing them efficiently. Users edit them with different word processing software, causing problems while moving them to different software systems, for example from MS Word to FrameMaker, LaTeXto WordPerfect, etc. In such circumstances, it would be very helpful to have a common format that is independent of the operating system and word processor software on which the document was originally created. All the documents could then be converted to this form and accessed easily. In this report, we describe different approaches possible to develop and establish such a format.

Section 2 discusses the access system necessary for a large database. Section 3 talks about the electronic page image approach for storing the document electronically. Section 4 describes the markup approach taken for developing an independent document format. Section 5 describes future work.

2 Access System

In this era with abundance of information available, it is very important to have easy access to it. There are a wide variety of tools such as World-Wide Web (WWW), WAIS, Gopher, Mosaic, etc. available to access information in different areas. Users, for example students, may be interested in a particular area in computer science.

2.1 WATERS

Wide Area Technical Report Server (WATERS) is an ongoing project between Old Dominion University, Virginia Tech, University of Virginia and SUNY, Buffalo [6]. The report server at ODU is integrated with databases at the other three sites to make available their computer science technical reports. Figure 1 shows the query and retrieval functional diagram of the WATERS system.

Users access WATERS through WAIS. The source file for WAIS is maintained at ODU. The technical reports of each of the four universities are indexed at their respective locations and indexed files are sent to the ODU server. The files are indexed and sent periodically from all of the locations so that any change in the reports is updated at the server. When a user makes a query, the index files are searched and a list of documents is presented to the user. When the user selects a particular report, the server at ODU makes a call to one of the four sites whose report is requested. The report is then retrieved from the site and presented to the user through the ODU server. Later in Fall 1993, an alternate server will come online at Virginia Tech.

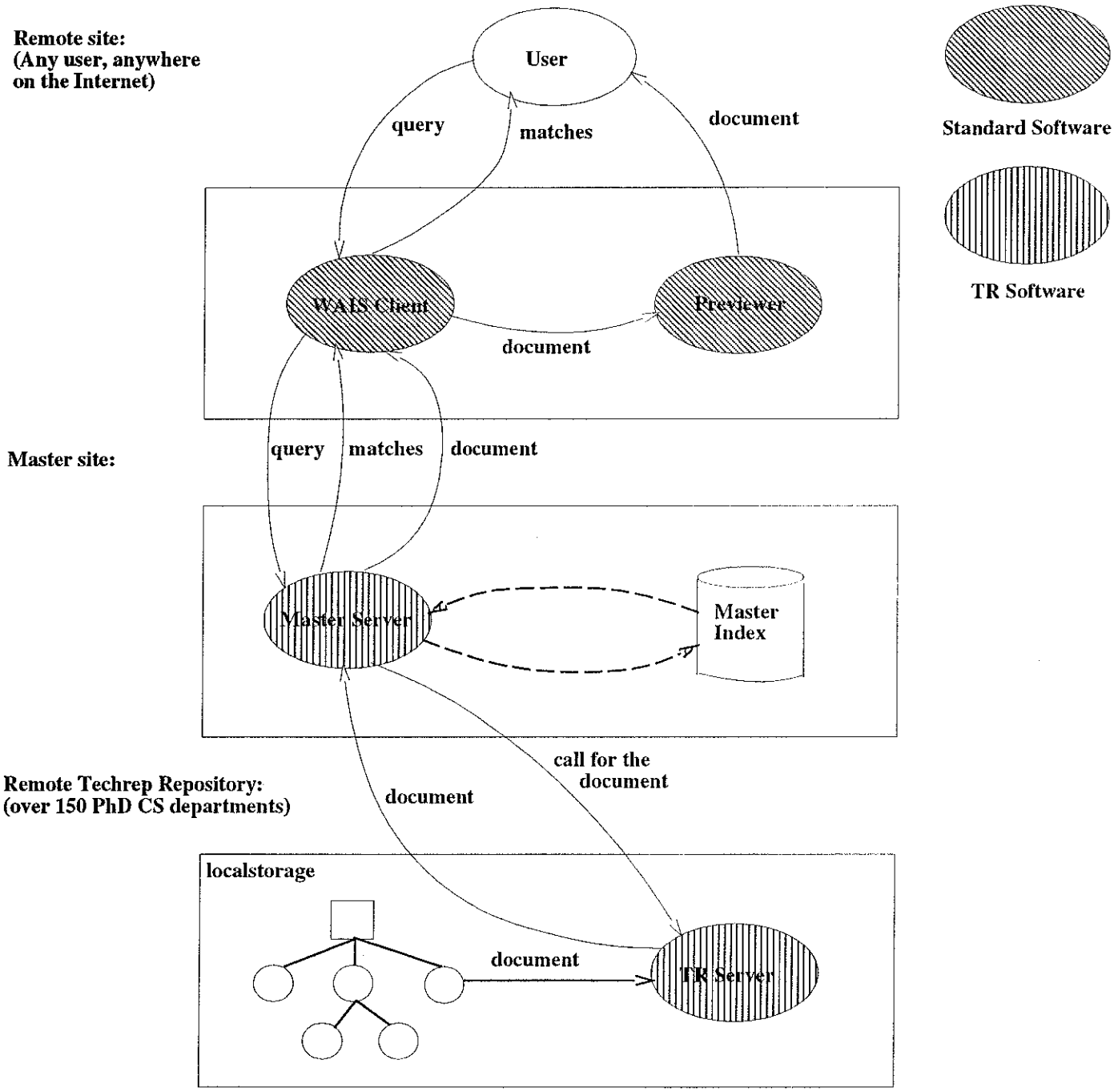

Figure 1: WATERS Distributed Indexing Functional Diagram

2.2 Techrep

Techrep is a stand-alone utility developed at the University of Virginia that has some of the same functionality as the WATERS system. Techrep has the facility to Post, Update and View the reports. However, users are not permitted access to Post or Update the reports. Only the local librarian has the right to change the contents of bibliographic information. Techrep is integrated with WATERS so that WATERS could be accessed from it.

Although at present, access to WATERS and Techrep is limited just to computer science technical reports; in future it could be extended to make technical reports of other departments and Master's and Ph.D. dissertations also could be made available through it.

There are two main approaches to represent such documents in a machine-independent format. The page image approach as discussed in Section 3 stores the documents in the Portable Document Format. Section 4 discusses an internationally recognized Standard Generalized Markup Language (or SGML) approach to represent such documents.

3 Page Description Language Approaches

Much of the difficulties computer users face could be attributed to the variety of computer platforms, operating systems and application software used in creating documents. The documents that now can be shared amongst different computing resources have to be simple enough. Thus it becomes difficult to share graphics and special formatting elements, for example. Due to these incompatibilities, it becomes difficult for people to access the plethora of literature available world over.

3.1 Portable Document Format

Adobe Acrobat software is a family of products that make the document communication among various platforms possible. Acrobat defines a PostScript [1] standard based file format called PDF (Portable Document Format) that is independent of hardware, operating system or application used to develop the particular document. A PDF file can describe the document containing any information including color, text, graphics, tables, figures, etc. There is no limit on the number of pages or the level of complexity of the original document. Once created, PDF files can be stored in computer memory just like normal files and can be ported to other computers without any difficulty [7].

3.2 Description of Acrobat

Table 1 shows the type of user and how he could use Acrobat. For example, faculty members can use Acrobat Exchange to view the dissertations, add notes and links to it. On the other hand, Graduate School can store the dissertations in PDF format and would be using Acrobat Distiller or Acrobat PDF Writer.

[1] PostScript format (also developed by Adobe) is the existing display-independent page description language

Type of User	Functions Required	Recommended Product
Students	Viewing,	
	Adding notes/links,	Acrobat Exchange,
	Searching,	Acrobat PDF Writer
	Making PDF files	Acrobat Distiller
Faculty members	Viewing, Searching	
	Adding notes/links,	Acrobat Exchange
Graduate School,	Creating and	Acrobat Distiller,
Library	Maintaining PDF files	Acrobat PDF Writer

Table 1: User Reference Table

The key products in Acrobat on Macintosh, PC and UNIX platforms include:

- Acrobat Exchange: It allows the user to view the document and browse through it. Users cannot change the original content of the document, i.e. documents are read only. However, they can make annotations, create hypertextlike links and search a particular text. Acrobat Exchange allows users to create bookmarks and thumbnails within the document to facilitate reading [7].
- Acrobat Distiller: Distiller is the program that carries out the necessary document translation. It converts the input PostScript files into PDF files which could then be viewed with Acrobat Exchange [7].
- Acrobat PDF writer: A specially designed printer driver used to make PDF files directly from the application. It is an alternative to the distiller for word processors like MS Word, WINWORD, etc. All of the three products are available for Macintosh, PC and UNIX platforms [7].

3.3 Creating PDF files

PDF files can be prepared in one of two ways: either by using Acrobat Distiller or by using Acrobat PDF writer.

1. Using Acrobat Distiller: This is a two-step process in which the first step is to make a PostScript file from the application software in which the document was originially edited. These PostScript files are then input to Distiller to give PDF files.
2. Using Acrobat PDF writer: Users can choose the PDF writer from the menu that appears when they select print option in word processing applications. The PDF writer printer driver then takes the document and writes a PDF file on the disk.

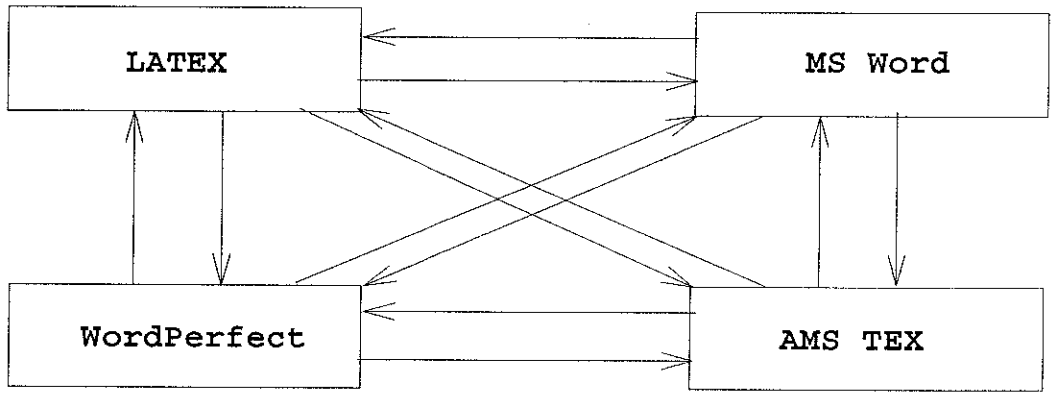

Figure 2: Number of Translators Required without a standard SGML format

3.4 Dissertations Converted to PDF

We have converted some of the dissertations to PDF format and they are listed in Appendix C.

We started with the installation of Acrobat Exchange and Acrobat Distiller. The first step of conversion is to get an access to the software in which the dissertation was originally edited. Possible examples of software in which graphic images were drawn are KXP, Harvard Graphics, Freelance Graphics, etc. Since the original document could be originally developed on any of the variety of softwares available, much of time and effort is spent in just getting access to the software.

Once the software is available, then the original document is converted to Postscript format. The next step of conversion is to give the Postscript file to Acrobat Distiller or Acrobat PDF Writer and convert the document to PDF format as described above. Large color bit maps can be downsampled to a lower number of bits per pixel or compressed using different degrees of JPEG (Joint Photographic Experts Group) compression. Once translated, documents could be stored and viewed with Acrobat Exchange.

4 Markup Approach

As there are different markup schemes and different software packages available in electronic publishing, it is very essential that there be a standard recognized format available. As shown in Figure 2, without such a standard, we need $n^2 - n$ translators (where n is the number of document formats) for translating each format to each of the other formats. By having a standard format, we can reduce the number of translators to $2n$ as shown in Figure 3. The first step of translation then would be to translate into the standard format (also called translation-up) and the second step would be to translate from standard format to the specific markup (also called translation-down).

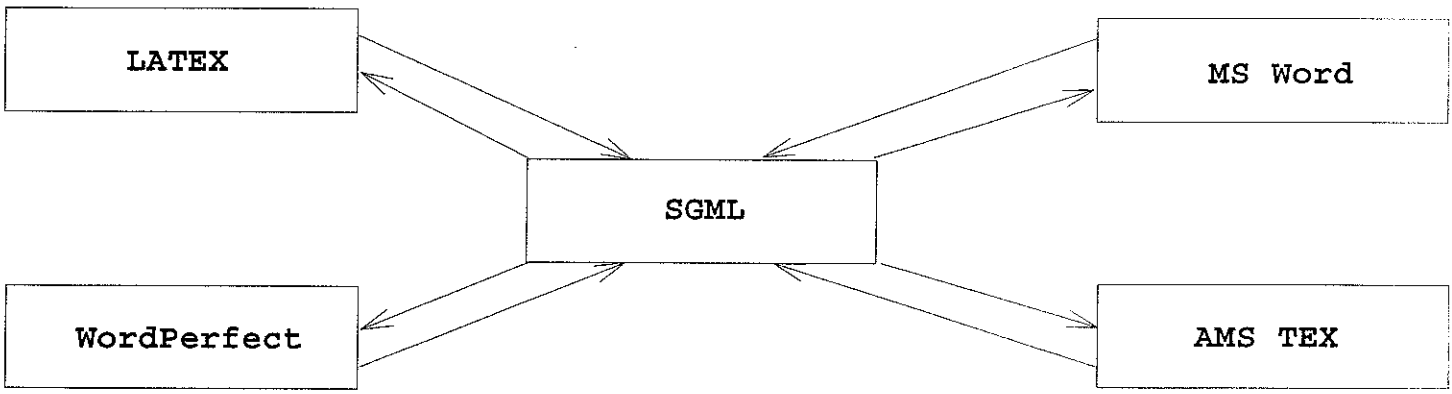

Figure 3: Number of Translators Required with a standard SGML format

4.1 SGML

The internationally recognized format for document markup is the Standard Generalized Markup Language (SGML). SGML expresses the logical parts of the document in a tree structure. Documents may be composed of chapters, chapters of sections, sections of paragraphs, paragraphs of words and words of character data [2] [3]. A hierarchical tree representation can be obtained for any document. SGML does not specify how to format or process the document, rather the structure of the document is embedded in the document text using *descriptive markup*. Descriptive markup is inserted using *start tags* and *end tags*. The text between matched pair of start and end tags is referred to as *content*. An element is composed of a start tag, content and an end tag. An element's content is described with a *content model*. A part of the structure for a dissertation is shown in Figure 4. A dissertation consists of frontmatter, body, appendix matter and backmatter. The frontmatter may consist of title page, publication information, abstract, table of contents, list of figures and a list of tables.

Following are the advantages of using SGML [2] [3]:

- Machine independence, meaning easy interchange of texts
- Generic markup, giving a separation of structure and layout
- Application independence, giving an open-endedness to data
- Unambiguous format, bringing benefits to authors and publishers
- Validation with SGML parsers.

A Document Type Definition (DTD) is an instance of a grammar that complies with the SGML rules [1]. A DTD must accompany a document wherever it goes, and programs called *SGML parsers* analyze and check that the markup in the document satisfies the rules defined by the DTD. A DTD defines the grammar for a particular style of document such as an article, a report, or a dissertation.

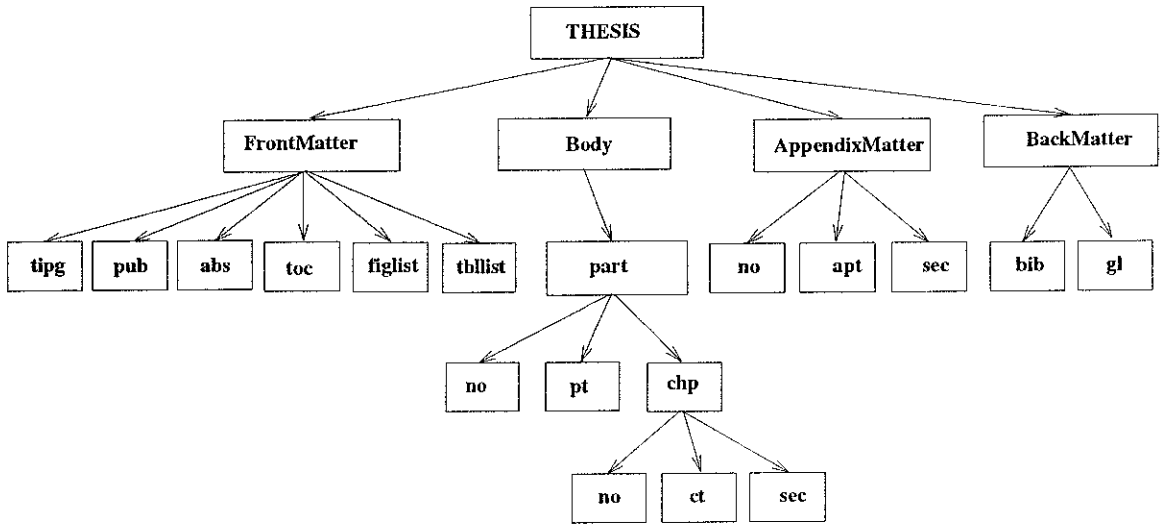

Figure 4: Hierarchical Representation of a Dissertation

Key:

abs - abstract	apt - appendix title	bib - bibliography	cht - chapter
ct - chapter title	figlist - list of figures	gl - glossary	no - number
pt - part title	pub - publication information	sec - section	tbllist - list of tables
tipg - title page	toc - table of contents		

4.2 Translation

The tool used for translation was the *Integrated Chameleon Architecture (ICA)* built by the Computer Science Department, Ohio State University. For documents of different type such as a report, an article, or a dissertation, grammar defining the document structure varies. ICA follows their respective grammar for each of the document. ICA executes a series of steps on the document before producing the final SGML document. If the original document did not comply with the specified grammar, one or more translation steps fail and no output is produced. ICA tools create *lex* and *yacc* files, generate C code from those files, compile them and build executable parsers / translators for the user to use. ICA also allows a user to develop the grammar or modify the existing grammar according to his/her requirement. ICA enforces completeness in the specification of the translators, and correctness in the translated data. Thus ICA helps develop complete and correct data translators. Following is a brief discussion of the tools used by ICA and how the conversion takes place [5].

4.2.1 Devegram

Develop Grammar (or Devegram for short) is used for developing, viewing and modifying the grammar. Users can define nonterminals, terminals or group terms along with the number of occurrences of each of them. Grammar for the document is defined in terms of a series of productions. Devegram allows the user to view the structure of the document either hierarchically or linearly (a series of productions). Once a .dgm file is ready, users can create an ICA specification file (.ica). Devegram also has an option for the user to make yacc, Backus Normal Form (BNF) or DTD files as desired [5].

Following are some of the productions for the *vpithesis* grammar.

$thesis \rightarrow fm\ bdy\ appm?\ bm?$

$fm \rightarrow G.52000*\ G.5A000*\ abs\ toc?\ figlist?\ tbllist?$

$G.52000 \rightarrow tipg\ pub$

$tipg \rightarrow ti\ sbt?\ au+\ deg\ dept\ sch\ comm*\ date$

$ti \rightarrow m.hd$

$m.hd \rightarrow G.1A000*$

$p.em.ph \rightarrow e1 \mid e2 \mid e3$

$e1 \rightarrow m.ph$

$m.ph \rightarrow G.2E000*$

$G.2E000 \rightarrow text \mid p.zz.ph$

$p.zz.ph \rightarrow q \mid pp \mid p.em.ph \mid p.rf.ph \mid p.txinl \mid emq \mid mathin$

$q \rightarrow m.ph$

$pp \rightarrow m.ph$

Key:

abs - abstract
appm - appendix matter
au - author
bdy - body
bm - backmatter
comm - committee members
date - date
deg - degree
dept - department
e* - emphasis elements
emq - emphasis in quotes
figlist - list of figure
fm - frontmatter
G.* - group elements
m.* - content models
mathin - within math environment
p.* - in paragraphs
pp - publication information
pub - publication information
q - quotation
sch - school
tbllist - list of tables
text - text elements
thesis - thesis
ti - title
tipg - title page

4.2.2 Retag

Replace Tags (or Retag) is the tool that recognizes the tokens in the original markup scheme that need to be replaced by tags in the general markup. The user specifies both the tags along with the context (if any) in which those tags appear. The tags in the specific markup scheme are divided in five categories [4] [5].

1. Symbolic Tags: Tags for mathematical symbols or entities such as λ, θ, β, etc.
2. Processing Instructions: Formatting instructions such as */newpage*, */hline*, */backslash*, etc.
3. Symbolic Tags: The symbol tags specified by the user for the general markup scheme such as $< P >, < SEC >$, etc.
4. Implicit Segmenting Tags: Tags for which in the specific markup only start or end tag (but not both) are present. Examples of such tags are /chapter and /item in LaTeX.
5. Explicit Segementing Tags: Tags for which in the specific markup both start and end tags are present. It's easy for a translator to detect these and replace them by the corresponding tags in the general markup. Examples of tags in this category are the document and itemize environments in LaTeX.

The user may have to build a series of translators using Retag to detect the tags. For example, detecting paragraphs in the original document has to be delayed until all other tags have been identified. After the specifications are given, the user selects *make* from the *code* menu to build the code. Retag then builds the executables for the user.

4.2.3 Intag

Insert Tags (or Intag) inserts the missing start or end tags in the partially converted file obtained from Retag. The tags that Intag needs to insert are the *implicit segmenting tags* in the Retag. The user gives the specification for those implicit segmenting tags in Intag's specification menu with information about which start or end tag needs to be inserted. The code is then built by selecting *make* from the code menu [5].

4.2.4 Spec2gen

The Specific To General (or Spec2gen) tool does the structure mapping of the converted document by walking through the tags in the document. It builds a .s2g file from the ICA specification file (.ica). The .s2g file has the order in which tags should appear in the general markup scheme in the converted document. If at any time it finds a violation of the structure of the document, the translation process fails and the user is prompted with an error message [5].

4.2.5 Gen2spec

The General to Specific (or Gen2spec) does the structure mapping of a document in general markup format to a specific markup format. In essence, Gen2spec tool is the inverse of Spec2gen.

4.3 SGML Conversion

We partially converted a dissertation to SGML format using ICA. The approach followed could easily be extended to convert the whole dissertation. The original dissertation was in LaTeX format. *Vpithesis* grammar style defines the rules and grammar to be followed in the dissertation. We followed this style for the hierarchical representation of the dissertation; we created ICA specification file to be used by Retag, Intag and Spec2gen tools.

We identified the LaTeX tags in the original dissertation and specified corresponding SGML tags to be replaced in the Retag specifications. The regular expression corresponding to the SGML tag and the context information were also specified. In Intag, we identified each tag to be either an implicit or an explicit tag. Also, the contexts in which certain tags occur within other tags was specified. ICA then checks to see if each of the inserted tag follows the *vpithesis* style. ICA also inserts missing tags where necessary and completes the conversion. The structure mapping file was generated using Spec2gen. The next step before printing out the translated SGML document is to walk through the converted file obtained from Intag and map the structure with *vpithesis* DTD. If at any point, Spec2gen finds an error in the converted document, translation process is aborted and no output results.

4.4 Perspectives

Table 2 gives several different perspectives regarding the electronic handling of documentation such as that of dissertations. It gives the different approaches and related issues with the Graduate School, faculty members and students. For example, following the Page Description Language Approach, Students may be converting their dissertations to PDF format and submitting it to the Graduate School. The Graduate School stores the dissertations and makes them accessible to students and faculty over the campus. Faculty members may view the dissertations, annotate it and make links in it. Similarly in the markup approach, students use translators to convert the documents to SGML format. Graduate School can republish or reuse the converted document.

5 Conclusion and Future Work

We followed both the Page Description Language and Markup Approaches to obtain a common machine-independent format. While the Page Description Language Approach is easier to use and is feasible at the moment, markup approach is difficult to follow and currently not practical. However, the advantage of the markup approach is that the documents could be republished and reused; Acrobat Exchange does not allow the users to change the document, although users can annotate it and create links in it.

Approaches	Graduate School/Library	Faculty	Student
Page Image Approach	Scan the documents Store documents	Add notes and Add links	
Page Description Language Approach	Database Distribute/Access	Add notes and Add links	Conversion from original to PDF format
Markup Approach	Republish Reuse	Add notes, Add links and Change the document	Use SGML and translators for conversion

Table 2: Different Perspectives for Electronic Documentation of Dissertations

There are several things that need to be done to make documents more accessible.

- A next step forward to document conversion would be to develop an electronic archive for all the documents, for example all the dissertations in PDF format and make them available to users all over the campus. Existing documents will then have to be converted to PostScript and then to PDF format. We need to make Acrobat Exchange and Acrobat Distiller available over the network and in open labs for people to use. Site license arrangements for these need to be discussed.

- Developing a translator for general markup scheme to a specific markup (for example LaTeX). Also, for translation up (that is, from specific to general scheme) the existing translator should be made more sophisticated to handle implicit tags appropriately and to handle all the LaTeX tags.

- Another possible area to be explored is writing documents using Author/Editor. This approach although practical for small documents is at least currently impractical for documents of the size of dissertations.

- Eventually the goal should be to develop translators at least for popular word processing software like MS Word, WordPerfect and Word for Windows. A related issue is to be able to convert a LaTeX file obtained by converting a MS Word document to LaTeX, and then to SGML format.

References

[1] SoftQuad Thesis.Rules, SoftQuad Inc., Toronto, Canada.

[2] Eric Van Herwijnen (1990), Practical SGML, Kluwer Academic Publishers, Netherlands.

[3] Goldfarb C.F.(1990), The SGML handbook, Clarendon Press, Oxford.

[4] Standard Electronic manuscript preparation and markup, Association of American Publishers, Version 2.0, 1987, Dublin, Ohio.

[5] Mamrak S.A., O'Connell C.S., Barnes J., Technical Documentation for the Integrated Chameleon Architecture, Department of Computer Science, Ohio State University, Ohio.

[6] Maly K., Fox E., French J., Selman A., Wide Area Technical Report Server, Technical Report, Department of Computer Science, Old Dominion Universiry, Virginia, May 93.

[7] Adobe Acrobat Products & Technology, An Overview. Nov. 1992., Mountain View, California.

APPENDIX

A MAC Environment

A.1 Acrobat Exchange

To use the Macintosh version of Acrobat Exchange, the system requirements are as follows:

- A Macintosh Plus, SE, Classic, LC, II, PowerBook, Centris, Quadra, or newer Macintosh
- Macintosh System software version 6.0.5 or greater
- 2 MB of application RAM
- An 800K or Apple SuperDrive floppy disk drive
- A hard disk drive with at least 7.5 MB of free space for installation; 3.5 MB of disk space is used after installation
- A PostScript printer or other Macintosh-compatible output device

A.2 Acrobat Distiller

To use Macintosh version of Acrobat Distiller, the system requirements are as follows:

- A Macintosh II, PowerBook, Centris, Quadra, or newer Macintosh (with a 68020 or later processor; a high-speed Centris or Quadra is recommended)
- Macintosh System Software version 6.0.5 or later (System 7.1 or later is recommended)
- 6 megabytes (MB) of application RAM (12 MB of application RAM is recommended)
- An 800K or Apple SuperDrive floppy disk drive
- A hard disk drive with 10 MB of free space before installation
- Acrobat Exchange or Acrobat Reader software

B PC Environment

B.1 Acrobat Exchange

To use the Windows version of Acrobat Exchange, the system requirements are as follows:

- A 386- or 486-based personal computer
- Microsoft Windows 3.1 or later
- 4 MB of RAM (8 MB of RAM recommended)
- VGA, SVGA or higher resolution display adapter supported by Windows 3.1
- A 1.44 MB 3.5-inch floppy disk drive
- A hard disk drive with at least 6 MB of free space (or 5.5 MB of free disk if you choose not to install the tour)
- A PostScript printer or other Windows-compatible output device

B.2 Acrobat Distiller

To use the Windows version of Acrobat Distiller, the system requirements are as follows:

- A 386- or 486-based personal computer (a high-speed 486-based or faster personal computer with a numeric coprocessor is recommended)
- Microsoft Windows 3.1 or later running in 386-enhanced mode
- 8 megabytes (MB) of RAM (12 MB of high-speed RAM is recommended)
- A 1.44 MB 3.5-inch floppy disk drive
- A hard disk drive with at least 5 MB of free space
- Acrobat Exchange or Acrobat Reader Software

C List of Dissertations

The following dissertations were converted to PDF format.

- Identification of Criteria for Delivery of Theological Education through Distance Education: An International Delphi Study - Gary L. Seevers, Jr., Department of Educational Research, April 93.
- Importance Sampling Simulation of Free-Space Optical APD Pulse Position Modulation Receivers - Kenneth Baker, Department of Electrical Engineering, April 93.
- Glyceric Response to a Peanut Butter and Cracker Snack in Non-insulin Dependent Diabetics and Nondiabetics - Anne E Glynn, Department of Human Nutrition and Food, April 93.
- A High Performance DSP Based System Architecture for Motor Drive Control -Milo D. Sprague, Department of Electrical Engineering, April 93.
- Perfect Hashing and Related Problems - Ramana R. Juvvadi, Department of Computer Science & Applications, June 93.
- Poisson-Lie Structures on Infinite Dimensional Jet Groups and their Quantization - Ognyan Stoyanov, Depatment of Mathematics, May 93.

Following dissertation was partially converted from LaTeX to SGML.

- Perfect Hashing and Related Problems - Ramana R. Juvvadi, Department of Computer Science & Applications, June 93.