Reengineering PDF-Based Documents Targeting Complex Software Specifications*

MEHRDAD NOJOUMIAN[1]
University of Waterloo, Canada
and
TIMOTHY C. LETHBRIDGE[2]
University of Ottawa, Canada

This article aims at reengineering of PDF-based complex documents, where specifications of the Object Management Group (OMG) are our initial targets. Our motivation is that such specifications are dense and intricate to use, and tend to have complicated structures. Our objective is therefore to create an approach that allows us to reengineer PDF-based documents, and to illustrate how to make more usable versions of electronic documents (such as specifications, technical books, etc) so that end users to have a better experience with them. The first step was to extract the logical structure of the document in a meaningful XML format for subsequent processing. Our initial assumption was that, many key concepts of a document are expressed in this structure. In the next phase, we created a multilayer hypertext version of the document to facilitate browsing and navigating. Although we initially focused on OMG software specifications, we chose a general approach for different phases of our work including format conversions, logical structure extraction, text extraction, multilayer hypertext generation, and concept exploration. As a consequence, we can process other complex documents to achieve our goals.

Key Words: Digital Libraries, Electronic Publishing, Improving User Experiences, Browsing Interfaces

1. INTRODUCTION

Published electronic documents, such as specifications, are rich in knowledge, but that knowledge is often complex and only partially structured. As a result, it is usually difficult for users to make maximum use of a document. The objective of this research is to develop an approach by which a typical published specification can be made more usable to end-users. We achieve this by reengineering the PDF version of a document in order to generate a new multilayer hypertext version of that document. This makes the knowledge more explicit, and facilitates searching, browsing, navigating, and other operations required by end users.

As a case study, we applied our approach to various OMG software specifications published in PDF format. However, we ensured that all aspects of our work are as general as possible so that the same approach can be applied to other documents. We chose OMG specifications because they (a) have particularly complicated structures, (b) are important to the software engineering community, and (c) have been studied in depth by members of by eeeto to0.002u(o(ar (o(are)

overcame these problems and generated a well-formed XML document with various types of meaningful tags, which facilitated our further processing.

Various techniques for text extraction: we experimented with numerous methods to create a usable multilayer hypertext version of the document for end users. We also applied the latest W3C (World Wide Web Consortium) technologies for concept extraction and cross-referencing to improve the usability of the final output.

A general approach for document engineering: although our targeted documents were OMG specifications, we chose a generic approach for various phases of our work including format conversions, logical structure extraction, text extraction, hypertext generation, and concept exploration. As a result, we can process other complex documents. We also established the major infrastructure of a document-engineering tool.

Significant values and usability in the final result: after showing how to create a more useful format of a document, we demonstrate the usability of our final outcome such as: better navigating and scrolling structure, simple textual content processing, efficient learning, faster downloading, as well as easier printing, monitoring, coloring, and cross referencing.

2. RELATED WORK

In this section, we review document structure analysis and some research with respect to analyzing PDF documents

Aiello et al. [2000] provide a framework for analyzing colored documents of complex layout. In this framework, no assumption is made about the layout. The proposed

[2005] introduce a new approach to explore and analyze a ToC based on content association. Their method leverages the text in

- Tagging structure: We prefer a fo a

tend to have consistent patterns of sentence structure and terminology in their document headings, various document body sections, and the index [Nojoumian 2007]. Our first assumption was that document headings (i.e., those that appear in the table of contents)

4.2 Second Implementation Approach

In the second approach, we developed a more powerful parser that focused on a keyword, LinkTarget, which corresponds to the bookmark elements created in the previous transformation. This keyword is attached to each heading in the bookmark such as headers of parts, chapters, sections, and so on. Therefore, as a first step, we extracted all lines containing the mentioned keyword and put them in a queue, named LinkTargetQueue. We also defined various types of headings in the entire set of OMG specifications with respect to its logical structure. This classification is shown in Table 1.

Table 1. Different kinds of headings

T	Sample Headings	Type 2
1	Part I - Structure	Part
2	7 Classes	Chapter
3	7.3 Class Descriptions	

5. TEXT EXTRACTION

Hypertext presentation has been a popular method for various computer applications dealing with large amounts of loosely structured information such as on-line documentation or computer-aided learning [Nielsen 1990b]. In this section, we take our

exactly where they have arrived. On the other hand, if the destination of a jump is an entire hypertext page, the above problem goes away.

- Less chance of getting lost: Users are less likely to get lost by scrolling in small pages in comparison to a long page. In a long hypertext page, after following a link, a user may then move to some other parts of the document. But then the user may not know how to go back to where they came from unless they happen to remember the section number or title of the section they came from. If instead the document is organized as many small hypertext pages, it becomes simply a matter of hitting the back button in the browser.
- A less overwhelming sensation: A smaller document should help users to manage larger amounts of information and understand the document more efficiently.
- Faster loading: Users are not always interested in downloading the whole document at once, especially when the document is fairly big.
- Statistical analysis: It may be useful to calculate the most frequent pages loaded and the time during which users stay in each page. This information could be used to improve the specification itself, and to determine what the most significant information is.

To prevent loss of the original order of a document, we created-7(n)5(i)25(e -4(-5(e1 .(b)-4.-5(ng)]TJ 0.002(

t

Since file names were created from the @Numberattribute, we were able to facilitate access to each of these files. For instance, by a simple piece of XSLT code, as shown in Fig. 3, we generated the related hyperlinks in the table of contents.

Fig. 2. Producing multiple outputs

Fig. 3. Generating hyperlinks in the ToC

In the next section, we illustrate how to connect these files together by Previousand Nexthyperlinks at the top of each page.

5.3 Connecting Hypertext Pages Sequentially

In the earlier section, we generated numerous hypertext pages for each OMG specification, for example, 418 pages for the UML Superstructure Specification. In a later

section, we will be creating contextual hyperlinks and the table of contents that will allow direct jumping to various pages. However, we would still like to link all pages together by creating Previous and Next links in each page. This will allow the reader to proceed through the document in its original sequence, should they wish to do that. Therefore, we first extracted all elements' attribute, named Number, sequentially (1, 2, …, 7, 7.1, 7.2,

it also created a folder, named images for the XML file. Adobe put all figures of the document in this folder, and named them as follows: folder-name_img_1.jpg to folder-name_img_n.jpg. The Fig. 4 shows the structure of the <Figure> element that has two children: (a) <ImageData> with its “src” attribute, and (b) <Caption>.

Fig. 7. List tag structure in the XML document

The child::* means select all children of the current node, and child::* [position()=1] means select the child which is in the first place, and so forth, as shown in Fig. 8.

Fig. 8. Importation of simple and nested lists

extracted contents of the <Name> element (e.g., Class) and the <Reference> element (e.g., StructuredClasses). We also linked this class to its relevant hypertext page by the <Subsection> element's attribute (i.e., @Number+html, for instance, 9.3.1.html).

Fig. 10. Part of tagging structures in the XML document

As an example, part of the XSLT code with respect to the extraction of the UML class hierarchy is presented in Fig. 11.

Fig. 11. UML class hierarchy extraction

We developed a simple script that could execute the above XSLT code repeatedly (plugging in each of the package names where Actionsappears).

7. CROSS REFERENCING

To facilitate document browsing for end users, we created hyperlinks for major document keywords (for example, class names as well as package names) throughout the generated user interfaces. As we mentioned previously, since these keywords were among document headings, each of them had an independent hypertext page or anchor link in the final user interfaces. These hyperlinks help users to jump from one page to another page in order to gather more information as required.

We developed the related XSLT code to produce required strings for keywords used in the cross referencing algorithm, Fig. 13.

Fig. 13. Producingrelated strings for crossreferencing

This code selects sections that consist of class descriptions, and then generates a string which is made from the following six substrings, for every class:

Name+@+Name+

For instance, Abstractionis a class name; therefore, its generated string is as follows:

Abstraction@Abstraction

We applied a similar approach to generate related strings for package names, for example, the following string is generated for the Actions as a package name:

Actions@Actions

As you can see, we isolated keywords from their corresponding hyperlinks by @ character. We also listed all of these strings in a text file, named UniqueKeywords.txt, and then executed the Procedure CrossRef() for cross referencing.

To generalize this cross-referencing approach for other keywords and documents, we simply extracted all headers (since each had an independent hypertext page or anchor link) with their corresponding hyperlinks in order to put them in the UniqueKeywords.txt file, and then executed the CrossRef procedure.

```
Procedure CrossRef()
    folder-name // a folder consisting of various hypertext files
    F // a hypertext file belonging to a document
    UniqueKeywords.txt // a file consisting of the mentioned strings
    L // e.g.: Abstraction@<a href="7.3.1.html">Abstraction</a>
    S1, S2 // string variables
    While (True) do
        F = Extract a new hypertext page from folder-name
        If (all hypertext pages are extracted) Then
            Break this while loop
        Else
            While (end of the "UniqueKeywords.txt" file) do
                Get a new "L" from the text file // a new line
                Split "L" into two strings from "@" character
                S1 = first part of the "L" // Abstraction
                S2 = second part of the "L" // corresponding links
                If (find S1 in F in one place or many places) Then
                    Replace All (S1, S2) // replace all S1 strings with S2
                End If
            End while
        End If-Else
    End while
End procedure
```

8. EVALUATION, USABILITY, AND ARCHITECTURE

In this section, we demonstrate reengineering of various OMG software specifications, and address usability of generated multilayer hypertext versions by comparing them to the original PDF documents. We also illustrate the architecture of a document-engineering framework with the reengineering capability of PDF-based documents.

8.1 Reengineering of Various OMG Specifications

For further evaluation, we selected wide variety of other software specifications from Object Management Group (OMG) webpage with diverse number of pages and headings. The sample result of this assessment on ten documents is demonstrated in Table 3.

Table 3. Samplereengineering of OMG specifications

Original OMG Specifications	Number of PDF Pages	Number of Headings	Headings Used in Cross-Ref	Number of Tokens in Doc Body	Number of Tokens in Headings	Data Analysis Results	Number of Hypertext Pages
CORBA	1152	787	662	13179	702	15.1%	788
UML Sup.	771	418	202	10204	378		

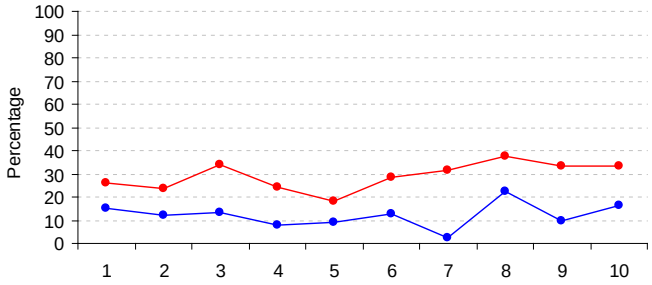

Fig. 14. Headings are among the most frequent words

All experiments confirmed that our approach is applicable to all kinds of documents. We just spent few seconds on some of these documents after the transformation phase to deal with rare mis-tagging problems. For example, forbidden notations among XML tags such as “>” (greater than) and “<” (less than) in some mathematical equations. Although this issue can be resolved automatically in our future design, the rest of our engineering procedures and software modules are totally automatic.

8.2 Usability of Multilayer Hypertext Interfaces

Heuristic evaluation is a systematic assessment of a user interface design in which a set of evaluators inspects the interface to judge its conformity with well-known usability principles [Nielsen and Molich 1990]. Nielsen [1989] compares 92 standard measurements of various usability issues related to hypertext in order to define those criteria that have the largest effects. Botafogo et al. [1992] also develop two types of metrics for hypertexts: global and node. The former refers to metrics concerning with the hypertext as a whole, and the latter focuses on the structural properties of individual nodes.

Although heuristic evaluation is not guaranteed to detect every single usability problem in an interface, this technique is a very efficient usability engineering method [Jeffries et al. 1991]. We first applied the same approach with the help of experts in our research lab. We then run a simple usability study among a group of software engineering students by designing multiple-choice questionnaires with an extra space for comments.

Our goal was to let them explore our user interfaces without any time limit such that they can also provide constructive feedback. For instance, they suggested that we add a Frame-like interface with a tree control on the left which shows the overall structure of a document, or create features that allow a user to add values to a document such as annotations, cross references, and links to related documentations.

In both methods, our intention was to compare the generated multilayer hypertext versions with the original PDF format as well as the HTML format of the specifications, which can be provided directly by Adobe Acrobat. This tool made a long hypertext page for each of those specifications along with anchors for headings at the top of each output.

Conklin [1987] summarizes operational benefits of hypertexts as follows: ease of tracing references, ease of creating new references, information structuring, global views in the ToC, customized documents, modularity, task stacking, and collaboration. Beside these advantages, we detected the following benefits through our usability studies, which did not exist in the original PDF formats, or Adobe-Generated HTML formats:

Fig. 15. Architecture of the implemented document-engineering framework

9. CONCLUSION AND FUTURE WORK

In this article, we described an approach for taking raw PDF versions of complex documents (e.g., specification

REFERENCES

AIELLO, M., MONZ, C., AND TODORAN, L. 2000. Combining linguistic and spatial information for document analysis. In Proceedings of RIAO Content-Based Multimedia Information Access, France, 266-275.

ANJEWIERDEN, A. 2001. AIDAS: Incremental logical structure discovery in PDF documents. In Proceedings of 6th ICDAR, USA, 374-378.

BELAID, A. 2001. Recognition of table of contents for electronic library consulting. International Journal on Document Analysis and Recognition, vol. 4, 35-45.

B

NIELSEN, J. 1990b. The art of navigating through hypertext. Communications of the ACM, vol. 33: 3, 296-310.

NIELSEN, J. 1989. The matters that really matter for hypertext usability. In Proceedings of the 2nd Annual ACM Conference on Hypertext, USA, 239-248.

NIELSEN, J. AND LYNGBAAK, U. 1989. Two field studies of hypermedia usability. In Proceedings of the Hypertext II Conference, UK, 29-30.