Internet-Based Teaching Using PDF-Documents

Harald Krottmaier
Institute for Information Systems and Computer Media
Graz, University of Technology
8010 Graz, Austria

ABSTRACT

The Portable Document Format (PDF) is very popular for distributing information over the Internet. It is often used as primary document format for lecture notes. Unfortunately there are many problems related to the format if it is used in context of an interactive teaching and learning environment. In such an environment it is necessary to personalize content for each user. Documents must be adapted to user's preferences. In this article we claim, that "ordinary" PDF is not optimal in that context. Documents must be enhanced. We take a look at features of PDF and will discuss problems when exploring the content of such a document. A simple solution for adding notes to PDF is presented. Since it is not possible to add notes to documents using freely available PDF-viewers we propose a server-side application to support readers of documents.

Keywords: Problems with PDF, Electronic Document Formats, Annotations in PDF

1 INTRODUCTION

PDF was developed by Adobe in the early 90s. Adobe was (and still is) well known in the field of electronic publishing and layout oriented document formats. The first popular layout-oriented document format developed by Adobe was PostScript. This format is still in use and after all those years still popular.

In this introducing section we take a look at the historic development of PDF to understand the intentions and enhancements of the format over the time. Thereafter we analyze the internal structure of the format. From the historical development it will become clear, that conventional PDF-documents are inapplicable in teaching and learning environments due to its strong *static behavior*. PDF was not designed to be manipulated by users. A short summary of available tools will close this section.

History

PostScript, an electronic document format — but also even a programming language for describing the layout of pages — was developed in 1984. The first version was known as PostScript Level 1. Beside text it was possible to describe graphic-elements in PostScript (which was not common in those days). PostScript Level 2 was published in the early 90s with performance and memory-management improvements[1]. Before PostScript was implemented in printing-devices the code was written by experts using some text-editor. Comments, operators (such as add and mul) as well as variables, procedures etc. were available to describe the page layout.

However, this layout-oriented document format was not intended to be modified. Since PostScript is a type of final output-format, modifications *must* be realized by some other program (e.g. the word-processor, desktop-publishing system or any other software) using the origin document format.

The successor of PostScript — PDF, the Portable Document Format – was developed in 1991 and brought to market in 1992. Internet at this time was still in its infancy.

PDF 1.1 (1994) was an improvement of version 1.0 introducing new features such as annotations and external links. Color-management and form-based features were introduced in PDF 1.2. PDF/X-1 (1998), PDF 1.3 (1999) and PDF 1.4 (2000) are continued enhanced versions of PDF. Mainly variations of paper-size, integrated security (i.e. digital signatures), integration of JavaScript as well as "tagged PDF" (additional information about the structure of a document and therefore additional features when displaying documents on different output-devices) are the main improvements. Currently PDF 1.5 (2003) is the latest version. Specifications of PDF are freely available at Adobe's website.

[1]The author is aware of many other improvements but since this article is not about PostScript but the successor of it, these improvements are not listed here.

The intention of PDF was to enhance *and* extend features available in PostScript. PDF is the "PostScript for the Web". On the one hand, text and graphics should be displayed in high quality, on the other hand the document should also be transfered to the user via the internet as fast as possible. PDF should also be displayed at user's side as fast as possible. Compared to PostScript some statements such as `if` and `while` disappeared. Therefore a faster rendering is possible.

Structure of PDF

Some of the features available in PDF were already mentioned in the previous subsection. However, to *understand* features let us take a look at the physical document structure. One may distinguish three different types of information in the document. Information about:

- object
- memory and
- document structure

We do not want to go into great detail of each of these types of information. One may imagine, that different subtypes of these information are specified. To give just a short example: Different subtypes of object-information are boolean, numbers, strings, names, fields, dictionary, streams etc. A very specific type is the stream type. Let us take a quick look at it.

Streams may be read incrementally. In addition to this aspect, it is possible to compress streams using different algorithms. Therefore it is possible to display parts of documents even before they are completely transfered to user's client. This is a very important feature in the context of an internet-environment.

To understand other features (such as annotations) it is necessary to take a look at the physical structure of a document (see figure 1).

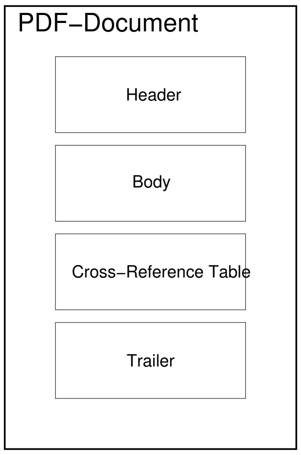

Figure 1: Physical Structure of a PDF-Document

header: there is exactly one header in a PDF-document. Any other part of the structure may exist several times. The header consists of just one line indicating the version of the document. To give an example:

```
%PDF-1.2
```

would indicate that the PDF-document follows the PDF 1.2 specification. Software for rendering PDF on screen require this information.

body: in this section the objects of the document are stored in a simple tree structure. An example is shown in figure 2. The *catalog-dictionary* is the root of the header, were single pages are stored (by means of different single objects). Thumbnails, annotation to pages, hyperlink-targets etc. are stored in this body.

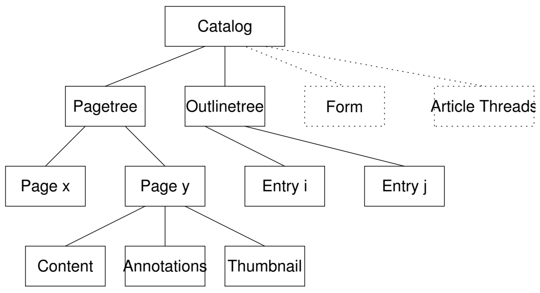

Figure 2: Header of a PDF-Document

cross-reference table: to make it easier to find objects, there is a reference table using 10 numbers (this limits the document to 10^{10} Bytes, i.e. about 10 GBytes). Objects are identified by object-numbers.

trailer: a special trailer-dictionary with references to important objects of the documents. This trailer makes it easier for applications to access objects of documents faster. The last line of the trailer is `%%EOF`.

PDF is not a structured document format such as XML. Parts of the document may be compressed to enable faster download, or the document may be encrypted. PDF is a very complex document format. It is layout-oriented and therefore it is not possible to address single words or paragraphs. When target-links are embedded in the document, it is possible to display directly the target when opening the document using special commandline-parameter (e.g. when using xpdf as PDF-viewer). Nevertheless, target-links (i.e. destinations) are not available per se. They must be created by the document author. Therefore the author must know in advance which parts of the document might be of interest to users.

Tools

As explained in the previous section, the structure of PDF documents is stored in form of a tree and nodes (partly in compressed form). Additionally reference tables and catalogs must be available for programs working with documents. Having this knowledge it is obvious that PDF can't be converted from existing document-formats, manipulated or created in an easy way *without* tools. Therefore tools were developed to make it easier to work with PDF documents. To mention just a few of existing tools:

PS2PDF: Many freely available tools allow a conversion between PostScript and PDF. Due to the similar features of these two document formats it is easily possible to write these programs. In the Windows-world tools such as Acrobat-Distiller ([4]) or Ghostscript ([5]) are available. Outdated versions of Distiller are available for Macintosh- and Solaris-platform.

creation of PDF: Within applications it is easy to print to files formated in PDF. Adobe distributes printer-drivers which additionally create links in the PDF documents. For LaTeX documents a PDF generator (known as `pdflatex`) is available. Additional packages (e.g. `hyperref`) create automatically intra- and extra-document hyperlinks.

programming / dynamic creation: Using programming libraries to create PDF-documents within the application is obviously the most powerful strategy. In web-application environments it is possible to create different documents for different users. A simple "customized document" can therefore be created when using this technique. Collections of programming-libraries exist for different programming languages. A very popular and freely available library is iText ([1]). This library is available for different programming language such as Java, PHP etc. An overview of other available (commercial and non-commercial) libraries is available at e.g. [2]. At the time of writing about 45 products are listed.

As one can see, there are many tools available for creating PDF-documents. Unfortunately, it is not easily possible to *work with* the document. Adobe seems to "forget" popular operating systems such as Linux or other Unix-based systems. On Windows users may buy Adobe's Acrobat (anyway, we can't expect students or pupils to buy software just to work with documents). For users of other operating systems than Windows it is not possible to *actively work* with the document, because there is no implementation of annotation-features, collaboration tools etc. available.

This section described the history and structure of PDFs. It showed that the structure of the document is too complex to be edited without the use of tools. In the next section we will give and overview of requirements of electronic teaching-material.

2 REQUIREMENTS OF TEACHING-MATERIAL

In this section we will discuss requirements for teaching-material from different points of view. We do not limit the discussion to a specific area of teaching. The presented list is not indented to be comprehensive (e.g. we do not cover aspects of intellectual properties, evaluation of courses, support of administrative tasks etc.).

Teacher's View

Let us here describe some requirements and aspects from the teacher's point of view.

easy to create: it must be easy for teachers to create teaching-material. Depending on the used software to create the material (i.e. word-processor) it is very easy using existing tools to create PDF-documents when appropriate software is installed on the system. When using LaTeX as source for PDF-documents, they may be created via `pdflatex`. This kind of software is freely available.

reuse material: an enormous amount of useful material is available on the internet in different electronic formats. Assuming we are allowed to include material from other authors: in PDF it is not possible to include dynamically content stored on other sites. In contrast to HTML[2]: when using HTML, authors of the HTML-page are able to use

[2]We do *not* claim that HTML is a proper format for teaching-material.

either Frame- or iFrame-Technology to reuse existing online available documents. Using XML as document format will make it possible to address even parts of existing material stored online (using XPointer-technology).

easy to distribute: nowadays nearly every university offers some kind of internet-based learning-environment. Distribution of content — very often stored in a single file — is very easy. When using multiple files (such as HTML files with images etc.) it is necessary to create a package.

Student's View

printable: Online material can't easily be read on the beach. Since learning-process takes place in different environments, it is very useful to print learning-material and take it wherever possible. In this case the limitation is the number of pages. It is not wise to print 1000 pages... Software to view and print PDF documents is freely available for different types of operating systems.

add notes: As already discussed in many articles (see e.g. [6]) it is useful to add annotations to electronic documents. Many existing learning-environments support the creation of notes to learning-material stored on the system. However, a printed document is different to an electronic one. Notes written with a pencil on paper are not transfered to the electronic version and vice versa. PDF viewer software does not allow to add notes! Adding notes transforms the documents to a kind of "personal customized" document. Highlighting a document, adding sketches or other marks are special types of notes.

discuss content: For students it is important to share ideas about a certain topic with others. Existing software (aka forums) may be used to discuss the content with other students, tutors or teachers.

To improve performance of students it is absolutely necessary for them to *add notes* to the material. While discussion about a document and sharing ideas are clearly tasks for the server-system used as teaching-environment, these tasks should be possible on the client-side. Unfortunately, it is not possible to add notes using freely available viewer-software. Forcing students or universities to buy Acrobat is no solution, since this software is just available to users using Windows operating systems. In the next section we take a look at a possible solution to this problem.

3 ADDING ANNOTATIONS TO PDF DOCUMENTS

PDF documents may be used interactively on the screen using hyperlink-features available in PDF. Many other interactive features are available in the format specification such as adding annotations of different types (sound annotations, text annotations etc.) or add some highlighting.

It is assumed that PDF-documents are viewed with free PDF-viewer software (e.g. Adobe's Acrobat Reader or xpdf [7]) by students. Therefore students can't interactively "work" with the document in terms of traditional manipulation of content such as highlighting important sentences of a document, or adding annotations to specific paragraphs, etc. simply because there is no free software available to them.

Since it is not possible to enable the annotation-feature on the client-side, we suggest to solve this problem at the server-side. That implies that users must be connected to the internet and must be authenticated to the system.

Adding notes to the whole document (i.e. no specific part of the document must be addressed when creating the annotation) should be easy to implement. Simply use a discussion server (with features such as private/group/public-discussion etc.) and discuss a document. This solution is fine for small documents and will be accepted by users. However, when documents consist of more than say 10 pages it is difficult to find the right note for the appropriate paragraph.

An acceptable solution to users is to add annotations to single pages of a document. It would be easier for users to find the target of the annotation.

We have implemented a prototype where it is possible to annotate single pages of documents. Existing PDF documents are manipulated at the server-side by the previously mentioned iText-library. A special hyperlink is added to the footer of every page. This link leads to a special-annotation feature implemented at the server-side. The user might then write the annotation, classify the annotation as question, answer, note etc. and submit the annotation to the server. At the server-side we use a Hyperwave-Information server ([3]) and create a special object (i.e. an annotation-object) with appropriate attributes. Since annotations are stored at the server-side it is possible to filter, share, and/or analyze annotations. Authors and many other readers of an annotated document may benefit from questions.

Annotations might not just be shared between users, but also between documents of the same content, but which are of different electronic formats. To give an example: we offer content in our journal in different electronic formats to the user (PDF, PS and HTML). Since annotation-objects are stored at the server-side it is possible to share them between the PDF and HTML-format of the content. A prototype simply added annotations of PDF as special PDF-annotation-objects to the corresponding page.

4 CONCLUSIONS AND FUTURE WORK

This article described some of the advances and problems of electronic documents stored in PDF. A very specific problem, namely adding annotations to those documents, was explored in detail. We argued, that annotations are a main feature in electronic documents and this feature must be available to every user in a learning and teaching environment. Therefore processing of annotations should be implemented at the server-side.

Future work is addressed to integrate the prototype into the production environment, to perform usability tests, and to make exchange of annotations between different document formats easier to use. A number of attributes on how to process annotations on the server-side will be added.

ACKNOWLEDGMENT

This work is supported by DELOS, a Network of Excellence on Digital Libraries (EU FP6, G038-507618).

References

[1] Lowagie B. and Soares P.: A Free Java-PDF library `http://www.lowagie.com/iText/` (2004/04/12).

[2] PDF Store: an extensive range of PDF software for creating, editing and delivering PDF files. `http://www.pdfstore.com/` (2004/04/12).

[3] Hyperwave `http://www.hyperwave.com` (2004/04/19).

[4] Adobe Acrobat Suite `http://www.adobe.com/acrobat` (2004/04/19).

[5] Ghostscript: An interpreter for the PostScript language and for PDF `http://www.cs.wisc.edu/~ghost/` (2004/04/19).

[6] Marshall (1997). Annotation: From paper books to digital library. In *ACM DL*, pages 131–140.

[7] xpdf: A PDF Viewer for X `http://www.foolabs.com/xpdf/` (2004/04/17).