PDF Mayhem: Is Broken Really Broken?

Juha Lehtonen
CSC – IT Center for Science
P.O. Box 405
FI-20101 Espoo
Finland
juha.lehtonen@csc.fi

Heikki Helin
CSC – IT Center for Science
P.O. Box 405
FI-20101 Espoo
Finland
heikki.helin@csc.fi

Johan Kylander
CSC – IT Center for Science
P.O. Box 405
FI-20101 Espoo
Finland
johan.kylander@csc.fi

Kimmo Koivunen
CSC – IT Center for Science
P.O. Box 405
FI-20101 Espoo
Finland
kimmo.koivunen@csc.fi

ABSTRACT

In this paper, we focus on the quality of PDF files. We are interested in errors that validators report during the validation process: how accurate are these errors and can we build easy workarounds to avoid or even fix these problems? We present our findings from a pilot experiment where we validated more than 200,000 PDF files from well-known corpora with different validators and found several thousand problematic files. We then devised a process of reconstructing the invalid files and analyzing the converted data. Our results show that there are potentially working methods for avoiding problems during the PDF validation and these methods can significantly reduce the workload for preservation specialists who are responsible for the quality of the data. Our further aim is to master and manage PDF validation so that we can build an automated workflow which is able to migrate most of PDF files to PDF/A files during the ingest of a digital preservation repository. To achieve this in reliable manner we need further studies to build on what we have presented here.

CONFERENCE THEME

- Sustainable digital preservation approaches and communities

KEYWORDS

Automated digital restoration, file format validation, PDF, PDF/A

1 INTRODUCTION

PDFs, and especially PDF/As, are considered to be good file format candidates for digital preservation, although the format has its documented issues [1]. In this paper our focus is not to evaluate whether PDFs and PDF/As are suitable for preservation. Rather we start by acknowledging the fact that these formats are very widely used, but also that invalid PDF files are quite common, according to our experience. How to deal with these invalid PDF files pose a huge challenge within the context of digital preservation. In this paper, we analyze and provide preliminary results on how the tackle the problem of invalid PDFs and their errors automatically and with very little human intervention.

The Finnish national digital preservation service for the cultural heritage sector has been in production since 2015 [2]. Following international recommendations within the digital preservation community [3], we manage file formats and a high quality of data by maintaining national recommendations for file formats acceptable for preservation and transfer to the national services[1]. PDF files (either PDF or PDF/A) are unsurprisingly among the allowed file formats in our national recommendations.

Various Finnish memory organizations (libraries, archives, museums) with different kinds of background are ingesting digital assets, including PDF files, to our service [4]. Obviously, we do not have any control on how they produce the material. Sometimes even the partner organizations do not have that control by themselves since stakeholders may be providing the digital assets to them. We have found out that some of our partner organizations are using tailor-made software for producing PDFs. These software are not primarily made for generating PDF files and they produce PDFs only as a “side-effect”. Some of these tailor-made software fail to produce valid PDF files. Instead, they produce files typically containing several systematic errors. However, it seems that even if the errors are systematic, they have only minor, if any, effect on the presentation of a PDF file. Perhaps the main reason for these minor errors is the technical complexity of the format (see [1]).

Our preservation service validates all digital objects in a Submission Information Package (SIP) during the ingest process and rejects the SIP if the validation of any digital object fails. We have found out that invalid PDF files are common reasons for

[1] http://www.digitalpreservation.fi/specifications/

digital object validation failures. Thus, it would be better to validate files before ingesting them and fix potential problems beforehand. Partner organizations may believe, however, that if PDF files can be opened using tools like Acrobat Reader, they are valid. That is why many of them consider the "pre-validation" unnecessary. We have a need to support our partner organizations by defining an effective process to minimize the amount of invalid PDF files sent our service for ingest.

We will show that sometimes simple/minor errors in the PDF files can be corrected automatically and fairly simply. Automatic correction is very important since manual correction is costly, or even virtually impossible, if the number of files is large enough (consider manually fixing a minor error in 100,000 PDF files). In many cases, using existing tools to reconstruct the PDF file can achieve a desired result. We set the errors that we cannot correct automatically aside for future detailed studies.

2 EXPERIMENTAL DETAILS

We created a pilot experiment of a simple automated PDF restoration process. Our goal was to test the effort required to restore a wide range of PDF documents. This pilot experiment is a first step towards understanding the errors and their potential mitigation in our own validation process and the data submitted to us. We will now present our test data, the software used in the experiment and the automated process.

2.1 Test Data

We selected two different PDF test sets for this study: JHOVE PDF Test Corpora, and Govdocs1.

JHOVE PDF Test Corpora[2] is a test set for JHOVE by Open Preservation Foundation[3] containing 90 PDF files. One is a well-formed and valid PDF document just containing the text "Hello PDF-world!" The set contains 88 erroneous files synthetically created from the valid one. Additionally, there is a minimal PDF containing only a version header and end-of-file tag (we chose to skip this file). The reason for selecting this dataset was to have a manageable set of known errors as a controlled set.

Govdocs1[4] is a document corpus by Digital Corpora site[5]. This dataset is very large and heterogeneous, containing nearly one million files of various formats. These files are collected from the *.gov* domain by using Google and Yahoo search engines. We used only the PDF files from this dataset and skipped a few of those, since they either were duplicates or reported to contain malware. This resulted in a test set consisting of 229,597 PDF files. The reason for selecting this dataset was to have a wide range of real-life examples instead of synthetically generated errors.

2.2 Validation Software

We selected three different software for PDF validation: Ghostscript 9.22, JHOVE 1.20.1, and Xpdf 4.00. Additionally, we used veraPDF 1.10.6 for PDF/A validation and Acrobat Reader and Pdf-diff tool for manual comparison.

JHOVE[6] is the only one of these software, which is stated as a validator, and it is capable of validating various file formats. JHOVE is created by JSTOR and Harvard University Library and is currently developed by the Open Preservation Foundation.

Ghostscript[7] is an interpreter for PS and PDF files, originally developed by L. Peter Deutch for the GNU Project in 1986. Artifex Software is the current developer of the software. To make Ghostscript work as a validator, we simply converted the PDF files to "None". Ghostscript examines the file in the conversion and if it finds errors, it prints them to the standard output during conversion. Since the result type is "None", it does not give an actual conversion output. We also used Ghostscript as a reconstruction software from PDF to PDF and as a conversion software from PDF to PDF/A.

Xpdf[8] is a software developed by Derek Noonburg since 1995. Some open source PDF analysis tools use the PDF toolset contained in Xpdf and we chose to use the pdfinfo tool from this toolset. Pdfinfo extracts metadata from PDF files and prints out found errors.

veraPDF[9] is a PDF/A validator developed in the PREFORMA project[10] by the veraPDF consortium, whose members are the Open Preservation foundation, the PDF Association, and the Digital Preservation Coalition.

Pdf-diff[11] is a small Python tool for visual PDF comparison. It creates an image of the differences from two given PDF documents. *Acrobat Reader*[12] is a well-known reader for PDF files created in 1993 by Adobe Systems.

2.3 Validation and Conversion Process

Figure 1 depicts our validation and conversion process containing five steps (a)-(e). We validated each set with JHOVE, Ghostscript and Xpdf (step a). If any of these gave errors or warnings, we considered the PDF file problematic[13]. If the validators did not report errors or warnings, we considered the PDF file fully well-formed and valid. Since we were only interested in problematic files in this test, we skipped the well-formed and valid files in the following steps.

The next step was to convert the problematic files to both PDF[14] (reconstruction) and PDF/A-1b, see step (b). We used Ghostscript for this purpose and used the instructions found in Ghostscript's documentation when creating the PDF/A files[15]. Ghostscript actually rebuilds the PDF file and it uses various automated repair functions when doing so. The aim was to see

[2] https://github.com/openpreserve/jhove/tree/ipres/pdf-test-all/test-root/corpora/ipres-paper-pdfs/modules/PDF-hul
[3] http://openpreservation.org/
[4] https://digitalcorpora.org/corpora/files
[5] https://digitalcorpora.org/
[6] http://jhove.openpreservation.org/
[7] https://www.ghostscript.com/
[8] http://www.xpdfreader.com/
[9] http://verapdf.org/
[10] http://www.preforma-project.eu/
[11] https://github.com/JoshData/pdf-diff
[12] https://get.adobe.com/reader/
[13] To be exact, it is not obvious if the error message appears due to an erroneous PDF file or an error in the validator software. Therefore, we consider both cases "problematic".
[14] gs -dBATCH -dNOPAUSE -sDEVICE=pdfwrite -sOutputFile=output.pdf input.pdf
[15] https://www.ghostscript.com/doc/current/Ps2pdf.htm#PDFA

which errors and warnings disappear after the conversion. To indicate the validity, we used all the validators once again on all the reconstructed files (step c). For the created PDF/A files, we additionally used veraPDF to validate the PDF/A features (step d) for the PDF/A files deemed valid in step (c). As a final step (e), we manually compared some of the original and reconstructed files with Acrobat Reader and Pdf-diff, specifically the files that the programs originally considered problematic but deemed well-formed and valid after the conversion step (b).

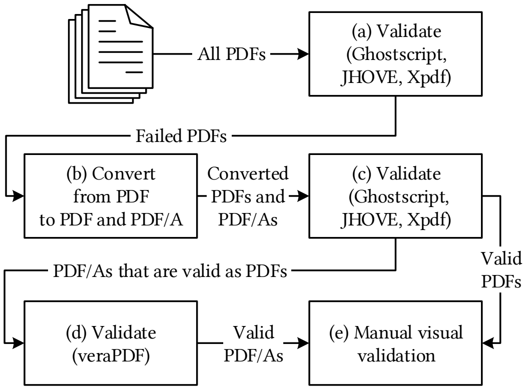

All PDFs
(a) Validate (Ghostscript, JHOVE, Xpdf)
Failed PDFs
(b) Convert from PDF to PDF and PDF/A
Converted PDFs and PDF/As
(c) Validate (Ghostscript, JHOVE, Xpdf)
PDF/As that are valid as PDFs
Valid PDFs
(d) Validate (veraPDF)
Valid PDF/As
(e) Manual visual validation

Figure 1: Validation and Conversion Process.

As an actual result from the process, we got various error logs: Error logs from the validators in step (a), from the conversions in step (b), from the validators in step (c), a veraPDF error log in step (d) and a manual error log in step (e).

3 RESULTS

We analyzed the results from the automated conversion and validation experiment by summarizing the prevalence of the errors, comparing different validation software and studying how well the PDF reconstruction and PDF/A conversion fixed the errors.

3.1 Controlled Testing

Benchmarking the JHOVE PDF Test Corpora is widely studied by *Lindlar et al.* (see [5]). We reconstructed and converted the erroneous 88 PDF files into PDF and PDF/A-1b. None of the three validators reported any errors in the resulted files. veraPDF reported only one error in one PDF/A file (T02-05-01_007_invalid-font-size-operator.pdf). The error was a real value out of range: “Absolute real value must be less than or equal to 32767.0”.

In the original set, Acrobat Reader could open 36 files rendering the text “Hello PDF-world!” with the original font. However, Acrobat Reader displayed an error message when opening some of these files, or the program fixed the file and asked for saving the document when exiting. The program displayed no such messages for the reconstructed PDFs or converted PDF/As. After the conversion, the text was visible in 45 files in the original font. For ten documents, the conversion restored the “Hello PDF-world!”, but in one case, the process lost the text (T04_018_trailer-no-xref-byte-offset.pdf). Additionally, in three original, four reconstructed and four converted files, the text was visible in a changed font. Acrobat Reader either opened the rest of the files in the original set displaying an empty document or did not open the files at all. The corresponding converted documents opened as an empty document, and only one of them did not fully render in Acrobat Reader. This was the same document that veraPDF reported invalid.

Overall, we could recover half of the synthetically erroneous PDF documents as well-formed and valid PDF/A files with the original text and font rendered correctly.

3.2 Prevalence of the Errors

The three validators in step (a) together deemed 30,346 PDF files in the Govdocs1 set problematic. This is about 13% of all the PDF files in the set. Ghostscript, JHOVE, and Xpdf found errors in 6,167, 20,595, and 9,116 files respectively. We got a total of 230 different error messages, with Ghostscript reporting 102, JHOVE 96 and Xpdf 32 different messages.

Table 1: Numbers of files and different errors.

	Original PDFs	Reconstructed PDFs	Converted PDF/As
Problematic files	28,313	3,528	2,647
Files skipped	2,033	0	8,317
Succeeded files	-	24,785	17,349
Number of files	30,346	28,313	28,313
Different errors	130	24	22

Table 1 summarizes the number of problematic files and errors. The reconstruction phase to PDF and conversion phase to PDF/A gave a conversion error for 2,033 files. We assume that these files were too broken to be handled easily and that the output most likely would not have been desired. Therefore, we moved these files aside and decided to restart the process for the remaining 28,313 problematic files. We decided to accept warnings during the conversion. The restarted process reduced the total number of different error messages to 130, with Ghostscript, JHOVE, and Xpdf reporting 29, 73, and 28 different messages for 4,187, 19,597 and 8,638 files respectively. Ghostscript and JHOVE reported errors on the same 415 files while Ghostscript and Xpdf reported errors on the same 148 files. However, JHOVE and Xpdf caught 3,576 same files. All three validators caught only 30 files. We can deduct from this that the validators do not find the same issues congruently.

After reconstructing the original problematic PDF files to PDF (not PDF/A), the number of different error messages dropped from 130 to 24, with Ghostscript, JHOVE, and Xpdf giving 7, 13, and 4 messages respectively. After the conversion, there were 3,528 problematic files left. A total of 24,785 (87.5%) well-formed and valid PDF files resulted from the originally problematic files. All the three validators catch 176 files. JHOVE and Xpdf largely find the same files problematic as Ghostscript, but JHOVE and Xpdf catch the same files only partly.

The conversion of the 28,313 problematic files from original PDF to PDF/A was skipped for 8,317 files, since the converter reported that it did not meet PDF/A requirements. We decided to move these files aside, and used 19,996 files. This is 66% of the originally problematic files. After converting the original PDF files to PDF/A, the number of different error messages was 22 for 2,089 files, with Ghostscript, JHOVE, and Xpdf giving 7, 13, and 2 messages respectively. Interestingly, the three validators do not catch errors in any same file. Ghostscript and JHOVE only find two same files problematic and the same for Ghostscript and Xpdf. Still, Ghostscript reports problems on 803 files. These validators do not validate PDF/A properties, so as a final step we additionally used veraPDF for the succeeded, but originally problematic, files. As a result, further 558 files failed, making the total number of problematic PDF/A files 2,647.

Overall, and mainly due to the lack of non-PDF/A-compliant conversion in some documents, we were able to use a simple conversion script to make 17,349 (61.2%) well-formed and valid PDF/A files from the originally problematic files.

3.3 The Error Messages

We list the error messages that appear in over 200 original Govdocs1 PDF files in Table 2. Additionally, we list the most common errors after reconstruction to PDF and conversion to PDF/A. The table includes the number of the original files, the reconstructed PDF files and the converted PDF/A files that resulted in the given message.

The most common error message, concerning 12,503 files, is JHOVEs: “Improperly formed date”. The other two validators did not detect this error. The three validators produced some other error message than the one above for 16,026 files, of which JHOVE detected 7,192 files. As we can see from Table 2, the most common error messages disappear totally after reconstruction and conversion. Some errors remain but the number of problematic files has reduced significantly.

However, a few new error messages (or increase in existing errors) occur after conversion. The Xpdf error message: “Bad annotation destination” in the original files actually changed to a warning message: “Unknown annotation destination type” in most cases. The JHOVE message: “Malformed dictionary” originally reported as “Missing or invalid default viewing OCCD” with Xpdf (OCCD = Optional Content Configuration Dictionary) in a majority of the cases. The Ghostscript warning: “File has an invalid xref entry … Rebuilding xref table.” usually is because of lost fonts: in some cases, the conversion did not embed the original PDF document’s fonts to converted PDF/A file due to licensing restrictions. If the font would be used in the document (that is, not only used in newlines or other non-printable characters), it would most likely result visual change to the document since PDF viewers use another font if the user does not have the original one.

Table 2: Number of files containing the most common error messages in Govdocs1 original PDFs, reconstructed PDFs and converted PDF/As.

Original PDFs	Reconstructed PDFs	Converted PDF/As	Validator	Message
28,313	28,313	19,996		Number of originally problematic files used in the validation
12,503 (44.2%)	0	0	JHOVE	Improperly formed date
5,090 (18.0%)	0	0	Xpdf	Syntax Error: Missing or invalid default viewing OCCD
3,446 (12.2%)	0	0	JHOVE	Invalid destination object
2,260 (8.0%)	0	0	JHOVE	...PdfInvalidException: Invalid destination object
1,906 (6.7%)	0	0	Ghostscript	openjpeg warning: Non conformant codestream TPsot==TNsot.
1,857 (6.6%)	0	0	Ghostscript	considering '0000000000 XXXXX n' as a free entry.
1,765 (6.2%)	0	0	Xpdf	Syntax Warning: Illegal annotation destination
1,751 (6.2%)	0	0	Xpdf	Syntax Warning: Bad annotation destination
653 (2.3%)	0	0	JHOVE	Invalid ID in trailer
406 (1.4%)	38	19	JHOVE	Lexical error
289 (1.0%)	238	13	JHOVE	Invalid character in hex string
242 (0.9%)	94	38	JHOVE	Invalid outline dictionary item
215 (0.8%)	213	38	JHOVE	Problem with page label structure
215 (0.8%)	0	0	Ghostscript	… Invalid indirect destination - referenced object '...' cannot be found
0	**1,634 (5.8%)**	723	Xpdf	Syntax Warning: Unknown annotation destination type
165	**736 (2.6%)**	236	JHOVE	Malformed dictionary: Vector must contain an even number of objects…
167	**634 (2.2%)**	226	JHOVE	Malformed dictionary
289	**238 (0.8%)**	13	JHOVE	Invalid character in hex string
0	**224 (0.8%)**	7	Xpdf	Syntax Error (…): Invalid hex escape in name
215	**213 (0.8%)**	38	JHOVE	Problem with page label structure
17	1	**757 (3.8%)**	Ghostscript	Warning: File has an invalid xref entry …. Rebuilding xref table.
0	1,634	**723 (3.6%)**	Xpdf	Syntax Warning: Unknown annotation destination type
165	736	**236 (1.2%)**	JHOVE	Malformed dictionary: Vector must contain an even number of objects…
167	634	**226 (1.1%)**	JHOVE	Malformed dictionary

We list the description of the most common failed PDF/A rules in Table 3. Rule 6.7.3-1 relates to property interpretation, Rule 6.3.6-1 to font rendering and Rule 6.2.10-1 to unrecognized operators in compatibility section. Proper handling of these errors may require expertise on the content of the PDF files. Therefore, these files warrant a separate investigation.

Table 3: The PDF/A rules where the dataset failed.

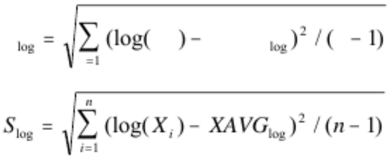
Files	Failed PDF/A rule description
237	Rule 6.7.3-1: If a document information dictionary does appear at a document, then all of its entries that have analogous properties in predefined XMP schemas, shall also be embedded in the file in XMP form with equivalent values.
202	Rule 6.3.6-1: For every font embedded in a conforming file and used for rendering, the glyph width information in the font dictionary and in the embedded font program shall be consistent.
63	Rule 6.2.10-1: A content stream shall not contain any operators not defined in PDF Reference even if such operators are bracketed by the BX/EX compatibility operators
56	Other failed rules

3.4 Manual Comparison

For the Govdocs1 set, we decided to select a few sample files from each of the error messages listed in Table 1 and manually compared the visual layout between document versions to determine whether some components in the document seemed to have been changed or not. We also used a PDF comparison tool to help with the manual comparison, but its results were not self-evident. The selected files were restricted to those that contain only one error and we selected files from each error group. We selected a few files randomly. Overall, 27 of the compared 42 documents were equal. Fifteen documents displayed the following changes: Selecting text became impossible in some parts of the PDF/A document but the visual outcome was equal (seven cases), a font was changed (four cases), extra features such as layers or security features were lost (two cases), font style (bolding) was changed (one case), and some pages were turned 90 degrees (one case). In some of the cases, the problem was in the process itself and not the PDF document. For example, losing a font is typically because of licensing restrictions. However, the loss of text selection ability probably was a consequence of an error in the document itself.

Additionally, we manually compared 30 randomly selected documents from all the problematic files and we found differences in only a few of them. These differences were practically of the same type as in the other manual comparison.

As seen in Figure 2, the conversion may give a different result as an outcome. In this case, it restored data: part of an equation was originally missing when handling the file with some non-self-correcting viewer. The example shown in Figure 2 looks obvious, but it may not be that in some other case. Govdocs1 is an external test set where we compared originally problematic files. We do not have fully well-formed and valid versions of these documents available nor sufficient provenance metadata (despite that we do have some short information available in the metadata of the file). Lacking knowledge of the content, we cannot be sure e.g. if the original erroneous file represents colors of an image or formulas correctly. Therefore, we decided not to go more into the details of the visual outcome.



$$_{\log} = \sqrt{\sum_{=1} (\log(\quad) - \quad_{\log})^2 / (\quad - 1)}$$

$$S_{\log} = \sqrt{\sum_{i=1}^{n} (\log(X_i) - XAVG_{\log})^2 / (n - 1)}$$

Figure 2: An equation from a Govdocs1 document shown in the Pdf-diff tool before (upper) and after (lower) the PDF reconstruction.

4 CONCLUSIONS

In digital preservation repositories where the amount of data is huge, there is a need to automate preservation activities as much as possible. Our experiment shows a potential way to handle PDF files and especially suggest methods to handle problematic files.

It is important to remember that the best way to fix problematic files is to focus on the process that produces these files. However, it is common for digital preservation repositories that submitted data is produced a long time ago and original data production may have ended. It might be impossible to reproduce the data or fix the production process which do not exists. For the repository, it is beneficial to recognize the really problematic or broken files and let preservation experts focus on these.

Our pilot study shows that reconstructing PDFs can be an effective method to reduce the amount of problematic files, but also that other “new” errors may appear this way. Out of 28,313 problematic PDF files, we were able to convert 87.5% to well-formed and valid PDF files, and 61.2% to well-formed and valid PDF/A files. Further studies are needed to understand how validators work in different situations and how to avoid unnecessary errors and problems in validation. Acquiring deeper knowledge about the meaning and importance of the errors is a pre-requisite for devising a process to mitigate the amount of invalid PDF files.

REFERENCES

[1] Marco Klindt. PDF/A considered harmful for digital preservation. In *Proceedings of the 14th International Conference on Digital Preservation* (iPRES2017) Kyoto, Japan, September 2017.

[2] Juha Lehtonen, Heikki Helin, Kimmo Koivunen, Kuisma Lehtonen, and Mikko Vatanen. A National Preservation Solution for Cultural Heritage. In *Proceedings of the 12th International Conference on Digital Preservation* (iPRES 2015), pp. 247-248, Chapel Hill, North Carolina, USA, November 2015.

[3] “File formats and standards”, in *Digital Preservation Handbook*, 2 edition, https://www.dpconline.org/handbook/technical-solutions-and-tools/file-formats-and-standards

[4] Juha Lehtonen, Kuisma Lehtonen, Aarno Tenhunen, Juha Törnroos, Ville-Pekka Vainio, Mikko Vatanen. Impressed by Ingest – Efficient and Reliable Workflows. In *Open Repositories 2016 Conference.* Dublin, Ireland, June 2016.

[5] Michelle Lindlar, Yvonne Tunnat, and Carl Wilson. A PDF Test-Set for Well-Formedness Validation in JHOVE - The Good, the Bad and the Ugly. In *Proceedings of the 14th International Conference on Digital Preservation* (iPRES2017) Kyoto, Japan, September 2017.