Open Planets Foundation (OPF) - PDF identify, validate, repair
1st-2nd Sept 2014, Hamburg, Germany.

A two day workshop/hackathon/mash up with talks from Olaf Drummer and breakout/hacking sessions investigating the different aspects of PDF and PDF/A formats and validation.

Twitter hashtag: #ofppdf

Event URL: http://wiki.opf-labs.org/display/KB/2014-09-01+Preserving+PDF+-+identify%2C+validate%2C+repair

Day 1: Monday 1st Sept 2014

Introduction: Yvonne Friese, Goportis / Ed Fay, OPF

See slides.

Introduction to the OPF as a member organisation made up of libraries, archives, commercial organisations, for knowledge sharing.

This workshop on PDFs driven by member need, there has been broad adoption - but there are many preservation risks.

Theoretically it is reliable, an open standard, has file integrity, is broadly adopted but there are questions about most of these things - mostly over preservation issues and longevity of format reliability.

PDF is becoming the most used format on the web and consequently in memory institutions

Broad risks involved with preserving PDFs:

- Identification - problems
- Validation - malformed files
- Repair - local policy conformance, validation, features - fonts, passwords, embedded files etc

Talk 1: Olaf Drümmer - Chairman of PDF Association - From PDF to PDF/A - ISO standard

See slides.

Olaf Drummer - background in PDF since 1996 has a company called Callas software - they made and licenced Preflight to Adobe.

What is PDF?

Background of PDF -what is it used for, what is the role of Adobe in PDF? Who owns PDF? - ISO standard so owned by ISO not Adobe. There are many tools which create PDFs, many developers worldwide and each creates a slightly different variation. What is the toughest part of PDF for developers? - fonts he thinks underlying issues from 80s and postscript.

Different versions:

- 1993 Adobe
- PDF 1.7 ISO 32000-1:2008 - iso time

- PDF ISO32000-2:2016 - PDF2 standard hoped.

Anyone can extend the PDF format

PDF - Imaging model

Unchanged since PDF1.0, it is very similar to postscript. There are some changes in the following versions - mostly to do with colour (colour calibration, transparency (1.4) layers (1.5) etc)

What exists on top of the page content:

- Annotations - links, mark up, forms, multimedia, 3D, digital signature
- Outlines - bookmarks
- Actions - javascript etc

Building Blocks of a PDF:

1. basic object types (names, strings, numbers etc)
2. resources - fonts, CMaps which bit of which font, images, metadata, ICC profiles, timed media streams, functions
3. filters - compression, encryption
4. text encoding for glyph look up (glyph from font)
5. text encoding from mapping to unicode
6. logical structure - tagged PDF similar to HTML 4 - headers, paragraphs, tables etc
7. Flash - issues - was important but seems to have been superseded.
8. XFA - issues - xml forms architecture - this is extra to the basic forms in PDF - allows branching according to question content.

PDF - it allows many things

What is best - freedom and richness/flexibility or reliability/predictability/interoperability

Now there are domain specific standards:

- PDF/X - print ready exchange files - no javascript/movies/encryption - good
- PDF/A - ISO19005 (2005-2012) - long term preservation of content in PDF files - good
- PDF/E - support for engineering workflows - almost no adoption
- PDF/VT - exchange of print ready for variable data, transactional printing (v slow adoption)
- PDF/UA - support for universal access in PDF files, readers and assistive technologies - becoming good
- XMP - extensible metadata platform lingua franca for metadata in PDF.

PDF/A is the only worthwhile digital paper archival format.

- for born digital static digital paper documents (e.g. word or 2D CAD files) – *not sure I agree with the CAD part!*
- for scanned documentation

Reasons:

1. PDF/A is self-contained - no dependency (fonts etc)
2. device, vendor, tool independent
3. unambiguous - text encoding and colour data
4. prohibits risky aspects - passwords, Javascript, casual handling of syntax

5. self-documenting - XMP metadata - not sure about this - only if the author actually wrote it
6. enables rich content representation (unicode, tagged PDF, object level XMP - eg. image description, copyright) - enables but does not enforce it.

What makes a good archival PDF:

- fully conforms with all rules in ISO 19005 (PDF/A)
- fully conforms with all rules in ISO 32000 (PDF 1.7)
- fully conforms with all rules in applicable referenced speci!cations and standards (font formats, image compression, ICC pro!les, Unicode, ...)
- contains images with sufficient image resolution
- scanned PDFs have been scanned well and contain error-free OCR-ed text
- all visual content is encoded/represented with perfect precision
- text is encoded as text and can be mapped to Unicode for useful text extraction
- content structure can be determined and used for 'rich' repurposing and migration
- visual rendering is always faithful to 'the original' and 100% consistent

Basically PDFs can adhere to the specification but not be very well done - OCR which misses things, images which are not of high enough resolution, does the table of contents reflect the content. Visual rendering is faithful to original?

Why care about archival quality PDF files if we don't have archival quality PDF viewers

Lots of readers absorb small errors and display things as well as they can - this is an issue with assessing archival quality PDFs - Adobe reader and FoxIT do this - as it is important for people to see the content but this is why we need a archival quality viewer.

So is the Adobe Preflight validation good enough? Or is this not falling under this umbrella.

Lots of developers work to a standard to mimic the display of adobe reader - this creates lots of buggy files.

How does PDF2 effect PDF/A - it will not change it but there might be additions… maybe PDF/A 2017 or PDF/A 4. There are no drastic changes mostly small changes which shouldn't make big problems.

Discussion

If PDF/A 1 doesn't work use PDF/A 2 - Olaf Drummer

Carl Wilson - discussion of how to work - dividing into groups of developers and practitioners

- Github - resource of all the OFP projects - devs need to upload their stuff
- Travis CI - continuous integration server. Checks that software works off a machine.

Testing tools afternoon with demo and workshop

This focussed upon…

Testing file identification and PDF/A validation tools.

Carl Wilson set up a virtual machine using Virtual Box and Vagrant which he used to distribute pre loaded versions of some common file identification and validation tools, these included, FILE, Jhove,

DROID for identification and PDFBox Preflight for validation. We were then able to test files sample files of our own using these tools. This was an interesting exercise and enabled participants to see the output of these tools and how that output could be used to populate technical metadata for the files. The file identification tools generally agreed with each other of the formats and versions of the PDF files tested. PDFBox Preflight did not appear to be the most robust tool as it was liable to crash if it was run on complex PDF files. It was not clear if this in itself showed the PDF was an invalid PDF/A or if it was just to large/complex for the software as it stands. It is however the only free open source validator available at present.

Break out groups

The afternoon was spent in break out groups, the groups were formed according to the common issues brought by the participants. The outcomes of the group work were written up on the OPF wiki (http://wiki.opf-labs.org/label/REQ/preservingpdf).

Investigating and assessing common PDF/a validation errors.

I had brought a number of common errors from PDF/A validation tools (mainly Adobe Preflight and PDFTron) and files showing these errors as my dataset for the workshop. I was interested in finding out which of these errors were 'important' errors and which were things that although not apparently easily fixed would not cause significant preservation risks in the future. We were fortunate to have Olaf Drümmer in our group and he was able to decode some of the common errors for us. The results of our findings and the work of the other groups is on the OPF wiki page for the session.
http://wiki.opf-labs.org/label/REQ/preservingpdf

Day 2 – Tuesday 2nd Sept 2014

Talk 2: What does it take to produce a comprehensive PDF/A validator - Olaf Drummer

Generally OD believes that PDF/A 2 is the most efficient standard to use (ISO 19005-2). PDF/A 1 is too restrictive and doesn't deal well with aspects of PDF which have become available since PDF1.4.

There are 17 specifications in PDF/A 2 and 79 in PDF 1.7 plus many more other variations that can be included. The difference between the ISO standard and the PDF specification is that the PDF specification is more descriptive and can be implemented incorrectly or less rigorously. This is because specifications often describe data structures without constraints so different people will implement things differently.

ISO standards aim to establish clear normative statements (shall, should, may) but they are usually written in prose and the format is very dense, lacking in guidance about how to actually implement it.

Adobe have developed a grammar for the design of the PDF format, as well as an object reference. This grammar (DVA Dictionary Validation Architecture) may become a part of the ISO standard. Adobe Preflight (validation tool) checks for syntax errors and performs these checks against this DVA.

Issues which arise when building validators

There are a collection of libraries and tools which developers can use for completing common computing tasks but when developers reuse libraries (JPEG etc) as the building block for validators they encounter problems. The libraries are generally meant for creation and manipulation but not designed as a validator. So they tend to be more flexible and are designed to cope with malformed files so that the software does not crash. For a validator however you want to highlight these issues and report on them rather than just absorbing them.

How do you ensure your validator works correctly:

1. You would have to build it without using any existing libraries (Callas software (Olaf Drümmer's company) has just done this for font recognition)
2. You then need to test it with test suites - files with known errors - isartor test suite - has existed since 2006/7/8 www.pdfa.org/2011/08/isartor-test-suite/ only done for PDFA1b (not PDFA1a or subsequent formats).
3. Aim for two or more independent validator implementations - two teams / two programming languages etc with no interactions - difficult in a commercial situation.

Commercial PDF/A validators:

Main vendors -

- Callas (Adobe, SEAL, FoxIT)
- Intarsys
- PDF tools
- PDFTron
- Solid Documents

Usual approach:

This is to check against things explicitly mentioned in the PDFA ISO standard. This gives implied validation - if the whole file can be read and it checks against all the ISO bits then it validates - however not all software reads the files in the same way and therefore not all the validators agree.

Practical value:

There is no monopoly so development is good - the vendors do communicate to match each other if a major new error is identified. The inference being that most validators will recognise similar errors even if they don't always do it consistently.

Economic side:

Validation is a largely commercial activity as no one will pay or volunteer (PDFBox - he's a little critical of how much work they can do as it is in their spare time etc) , however this might change - there are other projects - EU funding? PREFORMa project - a PDF/A validator - there are 3 formats that will be covered, yet to hear if the project will be funded. DPC is involved as well as the OPF.

If an EU funded validator is developed the software houses will support it as they can benefit as well.

Is there such a thing as a perfect validator?

XHTML W3C etc - good but not perfect - doesn't check Images etc.
Perfect validator doesn't exist as the perfect specification doesn't exist either.
XML validators work but still miss some complex things.
Code compilers are essentially validators but of code rather than of a format.

What should we focus on for validation?

He used a couple of examples which highlight issues of validation

Checking for embedded fonts

A table with true type font will have errors if it doesn't start with a 4 byte boundary (historic issue)

This is an issue with the true type font and has been inherited by the PDF and PDF/A - but should a validator highlight this - if it did report this it would be a massive issue and almost every file would be full of errors. *So which errors do we highlight and which are ignored.* It is an issue with font validation rather than the PDF structure but illustrates how it should fail on this factor. OD - Suggestion to rewrite the open standard font format to update it as this is an historic issue.

Page tree node problem – error reported by Jhove

This happens when the page tree is not balanced, but the specification for PDF 1.7 does not need a balanced tree - it works anyway (just with slower page access). It should be reported as information rather than an error.

A probabilistic view on assessing risk

- does it fail immediately
- how often is that feature used
- is it critical for using or preserving the file (text extraction - OCR issues)
- what is rarely verified - unicode mapping, metadata, colour encoding,
- what is not looked at often - metadata in the file incorrect content and syntax
- what risks are syntactical - should a validator check compressed content
- Which PDF syntax (and referenced technologies) features rarely used
- Which risks are on the quality / content levels
- what are challenging architectural aspects (values too high too low bezier curves) and big data aspects.
- Files with *interesting* problems - is it worthwhile to spend time on this?

Validating PDF readers

- Adobe reader hides many problems with files and displays what it can - we should find other more rigorous readers - he doesn't appear to have any recommendations on this?
- Major ones Adobe reader, FoxIT (Chrome agreement), Apple preview Quarts? since Mac OS10, Firefox engine (not so good).
- Apple preview and the console will give messages about issues exist with the PDF.

Discussion

Is it possible to validate a PDF by using a PDF/A validator and ignore any errors which are specific to the PDF/A spec. – no real answer.

OD - how to approach the PREFORMa project… a lengthy explanation

Next session:

- comparing different PDFA validators, callas, pdflib, pdftools
- Testing files using different identification tools - droid, fido, tika, file etc

Comparing different PDFA validators, callas, pdflib, pdftools

There is a webpage of different tools which will do validation (http://wiki.opf-labs.org/display/KB/Commercial+PDF-A+Validators+with+Trial+Functionality)

The problem file: wessexar1-151039_1.pdf

Up to date version of PDFTron works and says it can convert and validate to pdfa1b but it doesn't validate on Adobe Preflight PDFA1b (still has number out of range error)

So I tried to validate it against PDFA2b but that then popped up a CIDset error which had not even appeared in the PDFA1b validation - this doesn't inspire faith in the Adobe Preflight tool and one wonders if the PDFTron version is good enough? As it retains the vector images on some of the pages rather than the Adobe solution which was to change it all pages to raster images.

Comments from the end of the workshop:

- Surprised by the amount of different problems from PDFs
- Comparison tool to work out which PDF creator produce files which do not validate easily using PDFBox Preflight - no conclusions yet reached. They will add to the wiki.
- Conversion issue of dropped images in PDF-PDFA conversion - couldn't find a simple open source solution - having to via ghostscript and then to PDFA.

Details of these on the wiki http://wiki.opf-labs.org/label/REQ/preservingpdf