SCHOLARLY PUBLISHING OFFICE WHITEPAPER*

Proposals for a New Workflow for Level-4 Content

Kevin Hawkins

Executive summary

This paper describes the current workflow for SPO publications where the desired output is Level-4 text, covering the entire procedure from receiving text from content providers to producing files used by DLXS. After summarizing the disadvantages of the current workflow, it proposes two new workflows (the "Minimal Solution" and the "Grand Proposal") that would improve the content preparation process in terms of reliability and retention of data descriptiveness.

Current workflow

The current workflow is shown in Figure 1.

SPO receives source documents in various formats: XML, LaTeX, Word/RTF, and QuarkXPress. The first two are rare and have only occurred in one-off publications, whereas the others are used in regular publications.

These Word or QuarkXPress documents are first saved or exported in RTF format using standard DLPS filenames. We apply the **spostyles.dot** Word template to paragraphs in the document, attaching styles to them for the most commonly occurring content elements. The styling feature in Word only allows the user to label a sequence of paragraphs rather than build a hierarchy, but our conversion script attempts to build a bit of hierarchy into the document structure.

Then these documents are processed by R2Net v. 5.2.4 Server Edition, a command-line, proprietary, closed-source program by Logictran, Inc.[1] for converting RTF files to HTML or other SGML. When used with some scripts originally written by Brian Rosenblum and revised over time by Kevin Hawkins, R2Net outputs Text Class tagging in XML.

Our version of R2Net is a few years old; R2Net is now in the version 6 series. Furthermore, we are no longer paying for Logictran's "annual maintenance," which entitles us to free (and, in the past, helpful) support.

Metadata for the publications are created in header databases in FileMaker (one per collection), and these databses are exported whenever content is updated and reattached to bodies.

Problems with current workflow

Our current workflow has three main disadvantages:

1. The **spostyles.dot** template does not have styles for all content objects encountered in our source documents, so we fudge the tagging or make corrections by hand in the XML documents.
2. R2Net fails on certain files with footnotes, and after much trying, Kevin and Shana Kimball have not been able to determine what causes this or how to fix it without removing all footnotes.

[1] See http://www.logictran.net/.

3. R2Net v. 5.2.4 does not support Unicode, most non-Latin 1 characters do not convert correctly and therefore must be edited by hand in the XML.
4. We have no convenient method for generating PDF files based on source material.

Because we need to apply styles and occasionally tweak the source documents for unusual content objects and characters, revisions must be made in our XML files rather than in RTF files unless we reconvert a document from scratch. In this we deviate significantly from the ideal practice of reprocessing content only from the beginning of the process.

In addition, having separate databases for each publication has two disadvantages:

1. Revisions to the header structure must be made separately in each database.
2. Header encoding practice diverges without an authoritative way to encode.

The Minimal Solution (Plan 1)

Downfalls of the current workflow could be partially remedied by a few changes to the workflow:

Improving stylesheet

We could improve the inventory of styles in **spostyles.dot** and distribute an improved version of the template to our content providers, who could do the styling themselves. The style template would contain attractive formatting (including running headers and pagination) which the content providers would use for creating what appear to be camera-ready source documents. The template could be based on David Velleman's template, the MFR template, something designed by Julia Mitchell, or something that SPO creates from scratch. The file could be a Word template, an OpenOffice.org (OOo) 1.0 template, or an OpenDocument[2] template. The last would be readable by OOo and an increasing number of other software products, most of which are open-source. While Word is convenient because most content providers have access to it and are willing to use it, but we could set a good example by encouraging them to use open-source software like OOo.

SPO would receive files from content providers and correct the application of styles, if necessary.

Upgrading R2Net

We can upgrade R2Net to the latest version (providing Unicode support) and revise the conversion scripts so that conversion works smoothly enough for RTF files to be converted with little or no intervention and reconverted if changes need to be made to the source text.

Introducing PDF versions of content

We could provide PDF versions online for all publications rather than doing so only for those publications that give us with their own PDFs (like **mfr**).

Plan 1A

SPO could generate PDFs from the word processor documents (after SPO corrects the application of styles) by hand from within a word processor[3]. These PDFs could be delivered

[2] See http://en.wikipedia.org/wiki/OpenDocument.

[3] This could be done using the Acrobat plug-in for Word, using the printer driver included with OS X, or using a third-party product like pdf995.com.

through DLXS for personal use by users and also internally for setting up digital printing for all publications.

Plan 1B

We could use the XSL Stylesheets for TEI XML[4] to convert from TEI to PDF in batch using XSL-FO designed for the open-source PassiveTeX package. While this would take more time to set up than Plan 1A, it will give us a more scalable workflow. All of these components were created and are maintained by Sebastian Rahtz, of TEI and open-source fame, but he only recommends PassiveTeX for experienced TeX users who have an investment in it.

Plan 1C

Another option is to develop XSL-FO to convert in batch from Text Class to PDF. We would most likely use Sebastian's XSL Stylesheets for TEI XML as a basis for our XSL-FO stylesheets but need to test them carefully with another FO processor[5].

Building a unified header database

A unified header database of all item metadata would use all the fields in non-deprecated ways, and might even introduce enhancements we've been thinking of, such as encoding rights information and Creative Commons licenses.

This database can be built in FileMaker initially (because it's easy to manipulate while designing) but must be ported to a MySQL database accessible over the web, allowing distributed input of data from outside of SPO. There would be two disadvantages to continuing to use FileMaker:

1. A database stored on HTIWORK will be difficult for Mac users to access because of the Novell connection problem on Macs. The NetStorage web interface is good for downloading files but not for opening and modifying remotely.
2. A database hosted on the DSS server must remain in version 6 because DSS will not upgrade to version 7 or 8 due to security concerns. In this case, we will need to continue using Kevin's "escaped SGML" hack for handling non-ASCII characters.

Drawbacks

The Minimal Solution has two drawbacks:

1. We would continue to rely on proprietary software (R2Net).
2. We would give up hopes of ever archiving XML documents more granular than Text Class.

The Grand Proposal (Plan 2)

The proposed workflow is shown in Figure 2.

Instead of relying on R2Net for our conversion, we could use the OpenDocument format or the OpenOffice.org 1.0 format rather than RTF format as the source for conversion to Text Class.

SPO would create an OpenDocument template or OOo 1.0 template (but not Word template) like the one outlined in the Minimal Solution, only the styles would correspond more closely to TEI content objects.

[4] See http://www.tei-c.org/Stylesheets/teic/.
[5] Sebastian recommends XEP or Antenna House.

SPO would receive files from content providers and correct the application of styles, if necessary. Optionally, these OpenDocument files can be uploaded and made accessible through DLXS.

The OpenDocument files can then be converted to TEI and PDF.

1. The TEI OO package[6] of stylesheets has recently been re-released to convert from OpenDocument to TEI, either from within OOo or in batch with an XSLT processor. The TEI files could be uploaded and made accessible through DLXS (in addition to the OpenDocument files), presenting yet another alternative format for viewing and using the content.
2. The OpenDocument files can also be converted to PDF, providing another alternative format for download (in addition to OpenDocument and TEI) and allowing digital printing of more SPO content. In the short term, SPO could generate[7] PDF files manually from within OOo[8] (one option in Plan 1A, above). In the long term, SPO would adopt XSL-FO to generate PDF files (possibly multiple versions) from the XML in batches (as in Plan 1B or 1C, above).

Last, the TEI files are converted to Text Class in batch using an XSLT we would develop and maintain. It would most likely be based on a script developed by Chris Powell.

Note that these conversion processes, like the current R2Net procedure, are invoked manually (even in batch). Conversion does not happen dynamically in DLXS based on the source files, as it does with Text Class to HTML. Therefore, as the scripts or software are improved over time, the structure or appearance of newer converted content will inevitably differ from older content (as also occurs with content converted under the current system). This could be a problem even in non-serial publications, where a reconversion of the material (or a portion of it) at a later date could result in files differing in structure or appearance. However, as Brian Sheppard points out, we're likely to add exceptions in the conversion process over time rather than fundamentally change any conversion process, so this might not be a significant problem. The problem is not that this materially couldn't be fixed but rather that we can never guarantee that a file reconverted at a later data will yield the exact same output as before.

Article metadata could be stored in a unified header database (as in the Minimal Solution), or it could be stored in the OOo documents themselves. (See discussion below.)

This workflow has the following advantages over the Minimal Solution:

1. OOo is an open source program (as opposed to Word) that uses XML as the basis of its native document format. Version 2.0 can also read and write documents in OpenDocument format, a non-proprietary OASIS standard that has been submitted for ISO approval.
2. The conversion technology is likewise open source—based on XSLT, which is a programming strength of Jeremy Morse.
3. We would create archival-quality versions of the XML files (in a granular DTD) based on a well-accepted standard (TEI).

---

[6] See http://www.tei-c.org/Software/teioo/.

[7] SPO, and not content providers, should not generate PDF files because we want to have one source document (an OpenOffice.org file) from which we can generate TEI, Text Class, and PDF files automatically.

[8] SPO could use OOo's built-in feature, but currently there is no way to use this feature in batch mode or from the command-line without writing an OOo macro. Also, OOo's PDF creator creates very large files compared with those of other third-party PDF generating software like Pdf995.

The Header Question

There are two approaches to handling header data in the Grand Proposal. One is to adopt the method of the Minimal Solution: build a unified header database and continue attaching headers to bodies, as in the current workflow. A unified header database needs to be created for all the legacy content anyway, so we could easily continue using it for new content.

The other is to store header metadata in the OOo documents using OpenDocument's support for custom, arbitrary metadata. It should be stored in a granular format that will allow for conversion to, and from, TEI metadata. This method has the following advantages:

1. TEI and Text Class documents are produced automatically from the source files without needing to consult a database.
2. Source documents have all the content needed to produce other formats in the future. There is no need to reference an external database to retrieve the metadata.

There are some significant disadvantages:

1. If we need to change the header structure, we will need to reconvert all TEI documents using the TEI-to-Text Class XSLT.
2. OOo 2.0 only allows entry of four custom metadata fields. So we would need to create the metadata outside of OOo anyway.

These disadvantages to storing metadata in the OpenDocument files, coupled with the inconvenience of having old metadata in one system and new metadata in another, make it clear that we should continue storing metadata in a separate system. Unfortunately, this will keep us from considering OpenDocument a unitary archival format that includes metadata and content.

Even without using the metadata features of OpenDocument, we could still have content providers create metadata for us by reprogramming the Journal Management System (JMS) to fit the fields of the unified header database. The JMS could feed metadata into the unified header database and possibly, for archival purposes, the OpenDocument files or TEI files.

Note that the implementation of this Grand Proposal need not depend entirely on our current staff members because a replacement for Brian Rosenblum could help to implement it.

Plan of action for gradual introduction of the Grand Proposal

1. Kevin will talk with Alex Marsh about upgrading R2Net for now so that we have a smoother conversion process under the current workflow.
2. Jeremy will continue to develop the unified header database since it will be needed in any case.
3. We will develop an OOo template with attractive formatting. Initially this would be used for creating PDFs for uploading and digital printing, and OOo documents created with this template would be saved in Word format, opened in Word, and saved in RTF with the **spostyles.dot** styles applied for conversion using R2Net. When the other components of the Grand Proposal are ready, we will no longer need to work in Word at all.
4. Kevin will test the TEI OO package and look into making it work for OpenDocument-to-TEI conversion, first by consulting with the developer of TEI OO to see whether he plans to implement OpenDocument conversion.
5. Kevin or Jeremy will develop an XSLT stylesheet for conversion to Text Class based on Chris's stylesheet. We can test the new process on an **mqr** frontlist issue (since it has complex content), **jep** issue #2, the first issue of **plag**, or the next volume of **wsfh**.

Figure 1 – Current workflow

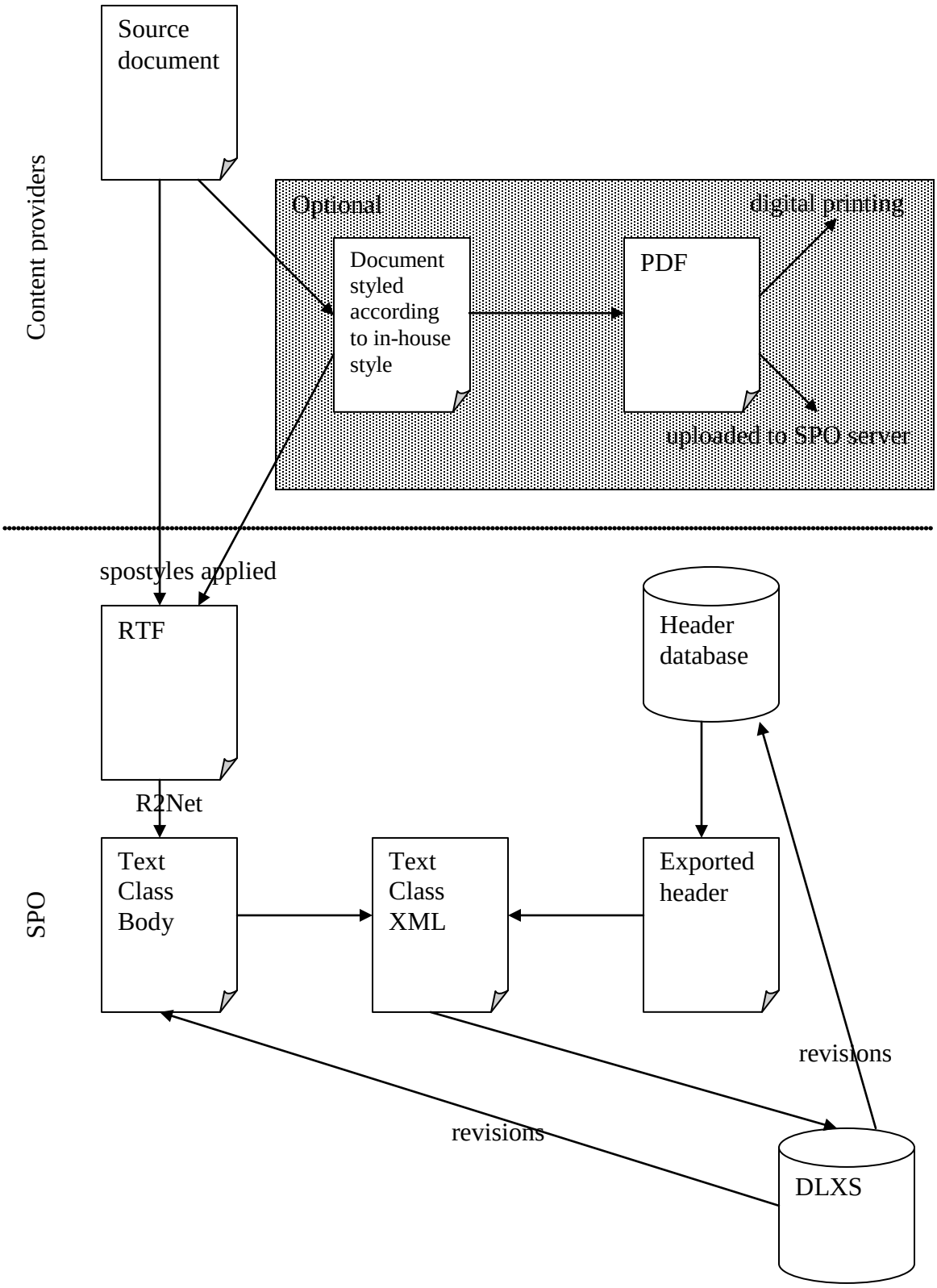

Figure 2 – Grand Proposal