The next wave in scholarly word processors?

This post is about the past, the present and the future, and also about word processors, PDF and that doomed Google Wave thing about which I have written a few posts.

Wave bye bye to the future

Cameron Neylon has posted some reflections on the impending demise of Google Wave, in which he saw great promise for the future of the dynamic scientific document, or whatever you call it. Document's probably not the right word for what Wave did, and it's not clear how document-like future scientific outputs will be. The start of his post is a lament about PDF and its constraints on scientific discourse; we're on the same page there. Wave, on the other hand, was dynamic:

> The key for me about the promise of Wave was its ability to interact with web based functionality, to be dynamic; fundamentally to treat a growing document as data and present that data in new and interesting ways. In the end this was probably just too abstruse a concept to grab hold of a user. While single demonstrations were easy to put together, building graphs, showing chemistry, marking up text, it was the bigger picture that this was generally possible that never made it through.
>
> http://cameronneylon.net/blog/the-triumph-of-document-layout-and-the-demise-of-google-wave/

Me, I won't miss Wave. I liked the idea of the document as a dynamic thing to which many parties could contribute, and I liked the way dynamic stuff could be plugged in. The dynamic parts were not all that much different to the OLE embedding that's been in Word processors for years, just repackaged for the web, but the idea of massively parallel editing by people and machines was exciting. What I didn't like was that Wave was actually a time-ordered conversation. You could branch the conversation at any point, as with a threaded discussion, but re-ordering was impossible. What I wanted to see, and still want was something with Wave-like collaboration but with well-structured HTML documents underneath. I'm with Cameron Neylon in thinking that we'll eventually figure out how to make these beautiful to behold (and **hold** – iPad style) and use and interact with.

I'll return to this at the end of this document,

Now to a contemporary critique of the past.

The present: word processors are (according to Sam) evil

In the past I have spent a lot of time and effort working on publishing systems for 'ordinary' users, focusing on word processor-driven web publishing. The latest of these, the Integrated Content Environment (ICE) is a core system at the University of Southern Queensland. I'm used to debating this core design choice of a word processor as an editing tool, and explaining over and over again why we chose Word and OpenOffice.org as the editors for our hundreds of users.

Actually, we didn't choose.

They chose. The users did.

Give them anything else with which to edit a long document and a significant percentage of contemporary

academics will type it up in Word, or increasingly Google Docs. A smaller percentage will type it up in any old text editor with LaTeX markup, and some other small percentage will use an HTML editor. Some gulp down the WordPress panacea.

Enter Sam Moffat, who works in the same division of the university as me, as a programmer, with his short piece: "The evils of a word processor". Sam reports that his mother is having difficulty with her word processor:

> I'm observing my mother fight with Word. To make things complicated she's copy and pasting items from a PDF.
>
> http://pasamio.com/2010/07/21/the-evils-of-a-word-processor/#more-669

The way I read this situation is that **the culprit here is actually PDF** not the word processor. PDF was designed to drive printers and to make it easy for publishers to lock documents to a particular set of formatting choices. Yes, I know PDF can be elegantly structured – but you know it rarely is. This is similar to what Cameron Neylon's saying; PDF constrains us. Pasting PDF into just about anything is going to be a hassle. (You did show mum how to paste-special as plain text, didn't you Sam?)

Sam's right, we do need to help our users get on with thinking about what they're writing and editing, and free them from the struggle to format things. And you know what? That's the intent of the ICE system; it uses styles in the word processor with an easy-to-use toolbar to apply them, and ICE users soon become comfortable with the way they can create a single source document – we don't turn off the other features of the word processor but we do try to wean people off them. But Sam's wrong about one thing, we don't 'trash' the formatting (see the next quote), we use the WYSIWYG word processor view of the document to make PDF. Why? Because that's a cheap way to render documents where all the disciplines and not a few of the people have special requirements.

It is nice to see that our programmers are thinking about the big-picture, productivity and the utility of our tools:

> I ponder this at times when the USQ course material publishing system went from a semantic based system into a system driven by OpenOffice.org and Word. I wonder just how much time our academics waste trying to get the format right when eventually it probably is going to be trashed anyway?
>
> http://pasamio.com/2010/07/21/the-evils-of-a-word-processor/#more-669

My judgement is that right now using a word processor to produce our single-source print and web materials probably wastes the least amount of time overall, taking into account the academics, support staff, trainers, systems developers, materials managers, publishing services people and so on, but I'm always on the look out for alternatives. We're exploring this as we consider what version three of ICE might look like. A SharePoint plugin? Part of a repository of quality-assured courseware? A Word Add-in?

One things is for certain, Wave wasn't it, and neither was a heavy-weight 'semantic' XML publishing system. We tried that.

The past: Why did USQ leave behind the 'semantic' publishing system?

Sam wonders about the demise of USQ's XML based 'semantic' publishing system.

It was called GOOD, but most of the academics regarded it as anything but. It came with an XML editor which was very complicated to use, not to mention hundreds of dollars per seat to license. There was very widespread resistance to its rollout. I remember the lead developer attempting to show me how he used it for the system's own documentation and giving up in frustration 'cos he couldn't really make the editor work. I also remember playing with it and trying to split a too-long section of text into two parts, with a sub-heading.

That's actually quite the gymnastic performance in a validating editor; creating a new section, getting the heading in the right element, moving the text, unless you know the tricks. Easy to do in a word processor, though, just whack in in a paragraph where you choose, and style it as a heading, relying on the well known fact that both machines and people can infer a perfectly good hierarchical document structure from cues given by different levels of heading – even if the headings are, *gasp*, put into table for formatting reasons..

And why am I putting those scare quotes around 'semantic'?

Well, there was the sad case of *work.title*. In GOOD, as in all good semantically-oriented publishing systems there was a plentiful supply of elements for describing things independently of the formatting used to display them in various media. GOOD had lots of ways of marking up stuff like the titles of works (someone wanted an element for the names of marine vessels, not sure if they got it – there were a lot of various expensive conversations about that kind of thing) and various kinds of emphasis. As you'd expect, more than one of these important semantic elements was rendered as italic in print. What happened was, the publishing services team, who worked on other people's manuscripts and wanted to get things **done** started using `work.title` for all italic formatting whether it was for the title of a work or for emphasis, or the name of a sailing ship. By the time people on the XML project team noticed, and more expensive meetings were held and so on, the only sensible thing to do was to batch-update all the documents so the polluted 'semantic' element became, you know, `<italic>`. (**NOTE:** this is from memory so the exact element names may be a bit wrong, but the story is true.)

In ICE we sidestepped this problem and we use the core inline HTML elements <i> and <b>. But, if ships are important to you then you can define a style i-i-frigate and format it how you like in your word processor, and ICE will give you `<i class="frigate"> … </i>`. Guess what? Nobody does that.

Sam concludes:

> As a LaTeX user, this is food for thought.

Um, right. LaTeX. I'm not sure what he's getting at here, as LaTeX is only semi-semantic and is often not really re-usable outside of the author's home directory.

Unfortunately, we haven't had much luck getting a LaTeX editing system that can produce both PDF and HTML for more than one person's output. Surprisingly there is no general purpose LaTeX to HTML converter on the web. I checked. More than once. And there is apparently no on-campus LaTeX user with the tenacity to (a) come up with a generically useful set of templates and (b) convince a community of others to all do things the same way. The quite tenacious Bron Chandler keeps trying to make this happen, though she's not a LaTeX user, working with a few of the less foaming-at-the mouth LaTeX users. Maybe Sam can help.

The future

Cameron Neylon liked Wave because it was not about formatting, and it was dynamic. I hated that it was not actually properly 'of the web' and it didn't make web documents; what I think we need in scholarship is **the web, but editable**, with ways to hook-in dynamic processes and declarative datasets that do interesting things when the dynamic processes get to them.

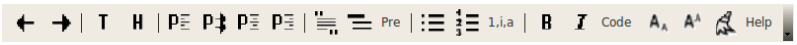
The good news is that Ron Ward, from our team at ADFI has been thinking deeply about this for a long time, and last week when the team had a 'choose your own adventure' innovation week, he chose to work on a pure-JavaScript in-browser editor. I know what you're thinking – there are tons of those already. Not so, according to Ron. Most of the editors you'll be thinking of are either not HTML editors (like Google Docs and Wave) or are using the in-built browser editing components. What Ron looked at was a real-life from the ground up 'line editor' for HTML something which apparently is widely regarded as being too slow and too hard. He had to build things from the blinking cursor up and did a remarkable job in a couple of days. Performance seems fine, and we're all really excited about the possibilities.

Ron's editor lets you click in an HTML paragraph, type away, changing the actual documents as you go . This real-live HTML has some interesting features. Type multiple spaces, for example and you only get one 'cos that's how browsers render space. The test rig is set up to capture editing events and replay them, opening up lots of possibilities for Wave-style time-slider interfaces so you can watch the evolution of a document.

Ron points out that parallel editing is not going to be that hard to do, if you set it up so multiple people can load the page at the same time. I haven't looked, but it could be that some of the theory behind Wave and some of the messaging framework might be useful here.

Ron's editor thing is really just a proof of concept at this stage, but things to explore include:

- Building-up the editor until it has an ICE-style toolbar like the one I'm using to write this document:



- Hook it up to our Anotar annotations framework so document-deltas can be stored as stand-off annotations. And there are conversations to have with Stijn Dekeser in USQ Maths and Computing, who researches in this area.
- Package it as an HTML 5 app to make a lightweight note taking thing for mobile devices.

I'll ask Ron to add a link to a demo/wiki here in the comments ASAP.

This post was written in OpenOffice.org, using templates and tools provided by the Integrated Content Environment project.