THE DIGITAL SCRIPTORIUM

Agile Methodologies and the Small Trade Publisher

by Kathleen Claire Lund Fraser
B.A., University of British Columbia, 2009

PROJECT SUBMITTED IN PARTIAL FULFILLMENT OF
THE REQUIREMENTS FOR THE DEGREE OF

MASTER OF PUBLISHING
in the
Canadian Centre for Studies in Publication
Faculty of Communication, Art and Technology

by

SIMON FRASER UNIVERSITY

Fall 2011

APPROVAL

NAME: Kathleen Fraser
DEGREE: Master of Publishing
TITLE OF PROJECT: The Digital Scriptorium: Agile Methodologies and the Small Trade Publisher

SUPERVISORY COMMITTEE:

Mary Schendlinger
Senior Supervisor
Senior Lecturer, Publishing Program

Rowland Lorimer, Ph.D
Supervisor
Professor and Director, Publishing Program

Susan Safyan
Industry Supervisor
Associate Editor
Arsenal Pulp Press
Vancouver, British Columbia

DATE APPROVED: 14 September 2011

Declaration of
Partial Copyright Licence

Abstract

THIS REPORT PROVIDES the technological, historical and theoretical context of an agile publishing workflow—that is, a workflow that adapts to a changing environment through iterative development. Given the current publishing climate, including the increasing pressure to produce print and digital formats simultaneously and the limited resources of small publishers, this report posits that such a workflow has become necessary. Building on ongoing research, it describes the structure of an accessible (low-cost and easily learned), web-first, agile workflow and the process of integrating it into the production processes of a small publisher, Arsenal Pulp Press.

The report concludes with a look at the system's successes and shortcomings, and the trajectory of the industry with the influences of proliferating formats and lengthening sales curves. In particular, it examines the changing role of the editor in the age of the short run and the integrity of the text given the flexibility and transparency afforded by digital approaches.

KEYWORDS: book publishing; independent publishing; agile publishing; digital publishing; ebooks; production models; editorial models

Acknowledgments

I AM GRATEFUL to my parents, Kelly and Dave, and my sister, Heather, who have been a constant source of support and inspiration. I am grateful also to my extended family and especially to Pam.

Many thanks to the team at Arsenal Pulp Press during my internship there, Brian Lam, Robert Ballantyne, Shyla Seller, Janice Beley, and my supervisor, Susan Safyan, whose patience, generosity, and editorial wisdom are boundless.

Thanks, too, to the faculty of the SFU MPub program, including Tom Woll, Roberto Dosil, John Maxwell (whose counsel throughout this process was indispensable), and my supervisors, Mary Schendlinger and Rowland Lorimer.

I am indebted to the remarkable 2009–2010 MPub cohort, and I am honoured to number in their ranks.

And thanks to the many friends without whose support I would be lost, including Michael Bround, Cynara Geissler, and especially Tracy Hurren.

Contents

Abbreviations

APP: Arsenal Pulp Press
CSS: Cascading style sheets
DTP: Desktop publishing
EPUB: Electronic publication
EWS: Editorial workflow system
ICML: InCopy markup language
IDML: InDesign markup language
IDPF: International Digital Publishing Forum
RTF: Rich text format
TCP/IP: Transmission control protocol/internet protocol
TEI: Text encoding initiative
W3C: World Wide Web Consortium
WYSIWYG: What you see is what you get
XHTML: Extensible hypertext markup language
XML: Extensible markup language

Introduction

PUBLISHING HAS ALWAYS been a business of adaptation. From the scriptorium to Gutenberg's printing press, from the industrialized printing process to Dent's paperbacks, publishers in every century have been faced with paradigm shifts. But at the same time, publishing is also a business of tradition (Steinberg 1996). Books serve many purposes, among them the preservation of knowledge and lore. Some books, and the information contained therein, have survived centuries. Publishers must now, as they always have, balance the instinct for preservation with the necessity for change.

Faced with the digital revolution, publishers must now find a way to integrate the tools and demands of the web into a set of practices rooted in the scriptorium, and honed with hundreds of years of technological advances. This report documents a small press's struggle to combine adaptability and integrity, in keeping with the long-running tradition of publishing.

In February 2010, six Master of Publishing students at Simon Fraser University (Vanessa Chan, Cari Ferguson, Cynara Geissler, Ann-Marie Metten, Suzette Smith, and Kathleen Fraser) set out to create a prototype of a web-first publishing workflow. Their stated goal was "to publish a book-length collection of essays in electronic and print-on-demand formats" (V. Chan et al. 2010, 3). By the project's end, they had "experimented with new technologies and explored the various ambiguities and unknowns that relate to them, ultimately documenting a state-of-the-art publishing project that serves to inform other publishers about the best practices for the production and marketing of single-source projects." *The Book of MPub*, as the text was called, was edited, designed, and released in four formats (print-on-demand, PDF, EPUB, and blog) in under ten weeks using technologies that are widely available to publishers. Ultimately, this project was a success on its own terms, but all those involved were technologically inclined students working in a low-risk academic environment. It remained to be seen whether the methodologies developed were applicable for existing trade publishers.

In the summer of 2010, the author served as the editorial intern at Arsenal Pulp Press with a primary focus on setting up a preliminary web-first workflow, training key staff in its use, and producing one digitally native book using the processes outlined by the *Book of MPub* team. This report is a record of that experience and a description of the Arsenal Pulp Press web-first editorial workflow system.

Section 1, on the print-first workflow, provides a brief history of twentieth- and twenty-first-century technology in publishing and introduces the status quo at Arsenal Pulp. It describes the incompatibility between a workflow that prioritizes print formats and the seamless production of digital formats, in the context of a similar workflow at Caitlin Press, another small publisher beginning the transition to new media.

The second section describes a new framework that prioritizes conversions rather than formats—that is, it posits that each conversion should take advantage of the content's initial state, in contrast to existing workflows that operate on a traditional format hierarchy regardless of complexity. Section 2 also provides definitions of key concepts including agile methodologies and extensible markup languages.

The process of introducing such a workflow at Arsenal Pulp is detailed in Section 3. It begins with the process of setting up an online editorial environment and the experience of editing in such an environment. The process of producing several final output formats follows. Section 4 provides an overview of the project's successes and failures.

In Section 5, the author speculates on the future of editing, given the inability of the web-first workflow to replicate the word processor-based editing environment. This section includes a discussion of the fluid text and a brief overview of the digital publishing models used by Wikipedia and OR Books.

The report concludes with a broader look at the economic, technological, and cultural implications of the web's increasing primacy as a site of not only communication and commerce but also creative endeavours. This report attempts to situate the agile publishing movement within this context.

1. The Print-first Workflow

THIS SECTION BEGINS with a brief history of digital publishing. It then details the existing workflow at Arsenal Pulp Press, from receipt of manuscripts to finished books and ebooks,[1] and the process of creating an ebook within a print-first workflow based on experiences at Caitlin Press.

Although much has been made of the rapid technological changes in publishing in recent years, the industry has adapted to many changes in the last century. The ebook may be a radical shift for the consumer, but the production process has been influenced by computerization since the first half of the twentieth century. Digital typesetting began in 1949, with the introduction of the Fotosetter, but did not reach its apex until the 1970s when computers became more widely available (IPC 2007).

The 1980s were an era of rapid technological innovation in the publishing industry. Packet-switching networks, in existence since ARPAnet's advent in 1969, gave way to the TCP/IP networking of today's internet by the mid-1970s (Salus 2008, Hauben 2004). In 1983, *Time* named the computer, now available to the public, the "Machine of the Year," and in 1985 a technology called Raster Image Processor was introduced, allowing the invention of the first desktop publishing software (Rosenblatt 1983, IPC 2007).

Today, desktop publishing remains the norm. Its introduction allowed publishers to bring typesetting and layout in-house. Essentially, it provides a WYSIWYG-only environment (using obscured, often proprietary markup) where publishers can manipulate layout, design, and content. Available desktop publishing (DTP) software includes Quark XPress, Adobe Pagemaker, and Adobe InDesign, part of the Adobe

1 When this paper discusses ebooks, it focuses in particular on the EPUB format, which is non-proprietary and is managed by the International Digital Publishing Forum. In the words of the IDPF, "EPUB allows publishers to produce and send a single digital publication file through distribution and offers consumers interoperability between software/hardware for unencrypted reflowable digital books and other publications" (IDPF 2011). For these reasons of flexibility and future-proofing, and because EPUBs are unencrypted bundled web content, this project aims primarily to integrate EPUBs into the existing publishing workflow. Proprietary ebook formats can then be produced from these files if necessary.

Creative Suite (Quark 2011, Adobe 2011). This paper addresses the design and production environments at Arsenal Pulp Press and, to a lesser extent, Caitlin Press, and focuses on InDesign, the software used by both companies.

Arsenal Pulp Press, like many publishers, operates with a print-centric workflow. Content is edited and designed for print production, which is to say that in the final copy, form and content cannot be separated: desktop publishing software like Adobe InDesign stores the text and its presentation in a single document. In the 1980s, when DTP first allowed publishers to integrate the manipulation of text with layout and design and streamline workflows, this was an elegant solution. Many companies combine DTP layout software with a word processor like Microsoft Word to achieve an entirely digital workflow, from editorial to production. InDesign does not provide an ideal editing environment, as it does not track changes or support comments, its files are large, and the software is relatively expensive and elaborate, if not inaccessible. Microsoft Word, conversely, is widely available and provides editing support but does not include robust layout features. Unfortunately, both programs use proprietary markup that is not particularly transparent or accessible to the user;[2] moving from word processor to layout software can sometimes be a cumbersome, convoluted process. The recent need for multiple, flexible outputs has complicated this process further. This section details the Arsenal Pulp Press (hereafter Arsenal) print-centric workflow, from receipt of manuscript to final print and digital outputs.

Arsenal requires that unsolicited manuscripts be submitted as hard copies. Once a manuscript has been accepted, the author is advised to provide it as a Word document, in double-spaced 12-point Times New Roman, without styles or inline formatting (Arsenal 2010). Generally, the book's editor prints this document and edits on paper. These changes are then made in the Word document using Track Changes; if a second editor is working on the same manuscript (fact-checking or formatting citations, for example), they work on separate copies of the document and then the changes must be collated. The edited manuscript is sent to the author by email.

The editor must also prepare the document for layout in InDesign. Before copy editing begins, all automatic formatting is disabled. Extra paragraph breaks

2 The 2010 IDML specification is 498 pages long; such a specification for Word's code has not been made public.

are deleted and alignment is corrected. Any Styles the author has used are deleted. The editor manually tags structural elements like headings and block quotes. Once copy editing is complete, the editor uses Word's Find and Replace tool to correct typographical errors including straight quotes and extra spaces.

Finally, the designer receives the document and places it into an InDesign document. InDesign paragraph and character styles partially automate the layout process. The designer typesets the text. Arsenal prints hard-copy proofs in-house and mails them to the proofreader; the designer makes corrections to the InDesign document. Finally, the finished file is packaged and sent to the printers.

Recently, it has become necessary to produce digital formats as well as print books. Arsenal must then extract digital versions from these final print-centric documents. Currently, the company relies on a freelancer. EPUB files are simply bundled XHTML[3] files, but it is arduous to transform content marked up with InDesign's markup language, which does not separate content from format, into a clean XHTML file and corresponding CSS file.[4] To date, the conversion process at Arsenal has been long, and attempts by the company to create ebooks that include images have failed.

Recent incarnations of InDesign have included an "Export to EPUB" function. The process began with exporting an EPUB from InDesign. Unfortunately, the print layout made use of some unlinked text boxes so the EPUB did not retain the correct linear order of the text. In addition, the resulting file was full of messy code. Brave is the ebook producer who will wade into InDesign's convoluted automatically generated CSS, and without including its comprehensive style definitions there is no way to preserve local overrides (spans of text in which one or more formatting characteristics differ from the default for the paragraph style; preserving local overrides is mostly relevant for italics in this paper). Exporting XHTML for Dreamweaver[5] (excluding style definitions) is a little cleaner, but not so clean as to be EPUB-ready, and using this export format means creating the EPUB structure in addition to editing the contents. In either format, the entire contents of the book are exported as a

3 Extensible Hypertext Markup Language, a simple structural markup language; see section 2.3 for further discussion of XHTML.

4 Cascading Style Sheets, a file containing formatting specifications associated with documents that use structural markup. See section 2.3 for further discussion.

5 A program for designing and editing web content. A part of the Adobe Creative Suite.

single file; it is easier to create navigation in ebooks if the chapters or sections are discrete files. The biggest problems with InDesign's EPUB output are the enormous and undiscriminating style sheet, including font information, and the use of local overrides, defined in the CSS file,[6] for italics and other localized style changes instead of inline HTML tags, which would allow users to attach new style sheets without losing functional formatting.

Although there is no comprehensive guide to best practices for EPUB formatting, *The Columbia Guide to Desktop Publishing* has this to say:

> Well-designed HTML documents achieve their impact from the simplicity of design; they are intuitive, coherent, and flowing…Especially when automatically generated from programs like word processors, HTML can get clogged with minutiae like elaborate use of fonts and fancy indenting that attempt to replicate a visual appearance that might not have been very good in the first place. Good HTML coding should not only produce a clear, clean page, it should do so with clean, well-structured tagging (Kasdorf 2003, 84).

Kasdorf's point is relatively intuitive: part of the appeal of an ebook is its flexibility. It must be readable on various devices with various screen sizes, and readers must be free to manipulate font size and style, margins, etc. The print-based design conventions do not apply to the ebook, and ebook readers are better served by creating files that remain attractive when reflowed at various sizes and in various styles than by attempting to replicate the fixed design of the printed page.

Examining an EPUB file exported directly from InDesign, this report's author determined that the style sheet would have to be pared down, the XHTML would have to be repaired accordingly, and the local overrides would have to be replaced with HTML tags manually. In addition, the book's contents would have to be divided into chapters for ease of navigation and to maximize the flexibility of the content.

For these reasons, it seemed easier to build the EPUB from scratch in dedicated software—a somewhat laborious process that could ultimately result in a more

6 That is, instead of describing a span of text as italic, the document assigns a "span class" to this span of text and, in the separate CSS file, specifies that this class should be displayed in italics.

functional and attractive file. The EPUB was therefore constructed in oXygen,[7] where each chapter was copied and pasted into a new XHTML file. The EPUB also included a basic external CSS file. Each text file was broken into paragraphs and spans of italics (which distinctions were lost when the text was pasted into oXygen) and each paragraph was tagged with the correct class at the same time. The file incorporated an image folder containing imported image files, to which the appropriate text files were linked; an XHTML table of contents; a content manifest; and other structural elements of an EPUB file. Finally, EpubCheck and EpubPreflight[8] (via ThreePress[9]) were used to validate the file.

Although this process was a success, it was both time consuming and frustrating. The text in the InDesign file had been through proofing and had been reviewed by the author before being printed, but the process of re-breaking paragraphs, manually marking up paragraph styles, and manually coding italics is a new opportunity to introduce errors. In addition, even if a small press had the expectation that ebooks would generate enough sales to warrant this much time and effort, it is not likely to have the staff resources and time to produce digital editions of all frontlist titles this way, not to mention backlist titles.

A better system would not require content to be formatted, stripped clean, and formatted again. Joost Kist, in *New Thinking for 21st-Century Publishers*, illustrates content production as variable processes—resulting in variable formats—applied to content, which exists independently. This ignores the fact that content exists in some state before these processes are applied, and because in the print-centric workflow it begins in a relatively inflexible state and becomes increasingly less flexible as it moves from word processing to DTP, the multiformat workflow in its current incarnation is linear—each format must be laboriously extracted from the previous. Because a document's formatting has, with the advent of desktop publishing, become integrated with its content, this formatting is difficult to strip out to produce cleaner, more flexible digital formats.

---

7 A powerful but proprietary XML editor.

8 Tools for determining the validity of an EPUB file.

9 A consulting company providing "digital tools for twenty-first century publishing." This includes easy access to validation tools as well as a browser-based EPUB reader. http://threepress.org/

2. A New Framework

THIS SECTION ADDRESSES the benefits of an agile, web-first/XML-based publishing workflow and provides an overview of a web-centric workflow system using free and open source technology.

2.1 WHAT IS AGILE?

THE *Oxford English Dictionary* defines "agile" as "1 characterized by ease and grace of movement. 2 mentally acute." In 2001, responding to the need to emulate such characteristics in a digital development environment, a group of seventeen visionaries drafted the Manifesto for Agile Software Development. This document is a list of four guiding principles for an agile development model—that is, a model that responds to rapid changes, both individual and environmental, through iterative development and minimal investment. These principles are described as hierarchies:

1. Individuals and interactions over processes and tools
2. Working software over comprehensive documentation.
3. Customer collaboration over contract negotiation.
4. Responding to change over following a plan. (Beck et al. 2001)

That is to say, in more explicit terms, agile processes respond to the unique needs of each user and creator rather than enforcing loyalty to rigid systems; such processes recognize that the development of software often outpaces its accompanying documentation and they prioritize relevant, applicable programs over blueprints and predictions; they aim to create relationships based on the strengths and needs of each party in each encounter rather than adjusting these relationships to fit boilerplate contract language; and they recognize that the challenges and opportunities a project will encounter are unpredictable and should be responded to appropriately at the time they arise, even if this means diverging from the initial trajectory of the project.

All of this means that an agile approach is bottom-up, i.e. composing a greater (and unpredictable) whole out of a system of unique encounters (between developers on the one hand and users, outcomes, successes, bugs, and so on, on the other). The creators of the "Manifesto for Agile Software Development" saw the development environment as top-down, creating a comprehensive plan before embarking on development and responding to each encounter according to the predetermined goal. Though there are many implications of these principles, the defining characteristic of an agile methodology is rapid, flexible, diverse prototyping (essentially, varied and extensive trial and error).

The agile manifesto has since been adopted by certain sectors of the publishing industry as an alternative to the top-heavy, inertial model built around the print-centric workflow. As digital formats and on-demand printing, which lend themselves to one-off editions, become more accessible, and as publishers move toward niche publishing and hyperlocalism, the economies of scale and advantages of conglomerates become less relevant, except in the production of bestsellers. Publishers have begun to take advantage of the lessons learned by the software industry by publishing on a smaller scale and using the inherent agility of the web in their production and marketing, theoretically paring down overheads and minimizing investment costs.

In this vein, Andrew Hunt, one of the authors of the Agile Manifesto, cofounded The Pragmatic Programmers (PragProg), a niche, software-focused publishing company that capitalizes on the technical abilities of its employees and authors in order to maintain an agile, cloud-based production model. Because PragProg is "by programmers, for programmers" and publishes mainly technical manuals, the skill sets and experiences of the authors and publishing staff are particularly well suited to a web-first workflow (Hunt and Thomas 2011). PragProg operates without offices or centralized equipment, and by using an XML-based system the company is able to reduce the staff time needed to generate multiple formats. Hunt and Thomas have claimed that this allows them to pay higher-than-average author royalties while remaining profitable (Rankin 2010). PragProg's advantages may be unique, but many small publishers strive for at least a portion of the portability and lightweight framework they have implemented. As digital formats are integrated into

the standard publishing workflow, agile processes become both more necessary and easier to grasp, as often the agile way is in fact the web way.

2.2 The web: simplest first

the web has existed since 1991, and best practices that have developed for future-proofing websites share all the defining characteristics of agile practices. That is, the web privileges those sites that are easily adaptable, well suited to collaboration and communication, and organic—in the sense that they evolve into complexity rather than beginning that way. Gall's Law applies here:

> A complex system that works is invariably found to have evolved from a simple system that worked. The inverse proposition also appears to be true: A complex system designed from scratch never works and cannot be made to work. You have to start over, beginning with a working simple system (Gall 1978, 71).

In the case of ebook production, this applies to both the format and, more generally, the shape of the workflow. With this in mind, a group of Master of Publishing students at Simon Fraser University prototyped a web-first workflow in the spring of 2010.

Drawing from the cohort's coursework for content, the team of six students produced a book in four formats over the course of nine weeks. Although the stated goal of the project was to "start with the web," in fact the articles were originated in Microsoft Word. They were posted to a WordPress[10] site, which served two purposes. First, it allowed the team to gather feedback and editorial insight from the publishing community at large, and second, it made use of WordPress's native semantic markup system (which describes the structural elements of a document in a way that is both human- and machine-readable), which is implemented through the Visual Editor

10 WordPress is a blogging platform and open-source content management system. It was used for this project because it is free, customizable, and supported by millions of users and developers. The size of the development community means many desired functions have already been developed and made available to the WordPress community as plugins.

and produces clean (valid, syntax-conforming) XHTML (see section 2.3 for more on XHTML). These advantages existed because the web already operates around communities, and uses as its skeleton a clean, lightweight,[11] extensible semantic markup language.

Once the book moved into production, the marked-up text was bundled into an EPUB file and simultaneously flowed into InDesign for layout. The team found WordPress's native markup sufficient for both tasks. At the end of nine weeks, the book existed as a live blog, a print book, a downloadable PDF, and EPUB. This outcome demonstrated that an agile production process could be successful in the context of a discrete academic project developed and used by students; the next step was to integrate agile approaches into an existing workflow at a trade publisher, which presented special challenges. Although many of the SFU team's discoveries were applicable for existing publishers, these companies face the additional challenges of learning and implementing new processes in place of the successful processes to which they are already accustomed while engaged in multiple simultaneous publications; factors like budgets, deadlines, and demands on staff are also different in trade environments from their incarnations in school projects.

2.3 What is XML?

To discuss the "web way" is to discuss XML and XHTML. XML stands for Extensible Markup Language. Per Kevin Goldberg, XML is "a specification for storing information. It is also a specification for describing the structure of that information" (Goldberg 2008, xii). Like other markup languages, it is a set of tags used to describe content to both machines and human users. The appeal of XML is that this set of tags is open-ended—the user can create and define tags as necessary. XML documents must also be well formed, following strict rules of syntax with regard to allowable characters, nesting, and document structure. The tag set is unlimited, so users can extend the language as needed, and the strict syntax means that as the

11 In this report, "lightweight" describes systems that eliminate redundancy and needless complexity. "Extensible" describes systems or languages that scale according to the needs of the project and environment.

demands on the language become more complex and the set of tags is expanded, the marked-up text itself remains valid and easily machine-readable. One branch of XML is extensible hypertext markup language—XHTML. In the words of the World Wide Web Consortium (hereafter the W3C), XHTML "is a reformulation of the three HTML4 document types as applications of XML 1.0" which is beneficial in several ways. First, XHTML uses the same syntax as XML and can therefore be read by XML tools like the popular XML editor oXygen, or any plain-text editor with built-in XML tools (like TextWrangler or Notepad++). XHTML documents are also usable in place of HTML documents (on the web, in EPUBs, and so on). Finally, because XHTML conforms to the same strict syntax XML does, it is more easily interpreted by programs in more environments than less strict markup systems. Future programs and markup languages are likely to be more forgiving to this strict syntax than to systems with limited tag vocabulary and haphazard syntax (W3C 2002). In general terms, this means that in contrast to HTML (defined by the W3C as "the publishing language used by the World Wide Web"), XHTML is better formed and therefore more flexible and future-proof.

In 2000, the World Wide Web Consortium began to recommend XHTML for semantic markup in web applications. Extensible Hypertext Markup Language is part of the web family, designed to accomplish the same tasks as HTML while incorporating both the extensibility and the well-formedness constraints of XML. WordPress uses XHTML, in combination with Cascading Style Sheets, and provides a user-friendly environment in which to apply both to content in the form of plain text.

Where markup languages describe a document's contents structurally and semantically (this is a title, this is a paragraph, this is a caption, this is the author's name), Cascading Style Sheets, or CSS, tell the browser how to display these elements (make all titles bold, highlight the author's name). Those who are familiar with desktop publishing or word processing software will see the parallel to styles, which allow users to define the qualities of a paragraph, for example, and then apply changes globally with ease.

The advantages of using XHTML in combination with CSS are clear. Content that uses inline styling (make these words bold, highlight these words) is static, specific to its context, and provides no structural information. Making changes to such

content, in terms of its appearance or platform, is cumbersome, especially in files like EPUBs and websites, which are composed of more than one file—while inline styles can be redefined globally within a particular document with relative ease, it is more difficult to do so over several documents, and EPUBs and websites rely on these document divisions for purposes of navigation. Semantic and structural markup, in contrast to the aesthetic markup of early HTML, keeps content apart from formatting. A document's structure is inherent, whereas its appearance is contextual, so XML is always relevant and readable, even if a particular edition of a text or the program in which it was created is obsolete.

2.4 Web-shaped workflows

existing publishing workflows are determined by two factors: priority of output formats (hardcover or paperback print formats, open or proprietary electronic formats, and so on), and merciless timelines. Small publishers (like most publishers) often have more work than staff resources so efficiency is key. However, this premium on time also means that even if an overhaul of the workflow would, in the long run, save time, there is rarely a block of uninterrupted time in which to implement and test new systems and processes. In the short term, it is faster and easier to integrate new demands and expectations into the existing workflow. Until recently, expected output formats have included hardcover, trade paperback, and mass-market paperback. Any and all of these formats were optional, and their releases were often staggered. Many publishers treat digital formats the same way: ebooks are seen as part of the format hierarchy and therefore optional and not urgent.

The format hierarchy, staggered release, and heavy demands on staff time contribute to a linear, seat-of-the-pants workflow. But a book is not its most recent format, and it is inefficient to pull each new edition out of the previous, especially now that expected outputs are not limited to print formats. Semantic markup makes content reuse more feasible by maintaining a master document that can be transformed as needed. EPUBs and websites use CSS, and InDesign uses style definitions, both of which rely on structural markup, which means this markup is best applied before production begins.

Small publishers must now find a way to use Joost Kist's approach—treating different outputs as different processes applied to the discrete body of text—without dedicating time and resources they don't have to creating and troubleshooting a new workflow. Such an approach is web-shaped in that content exists in a central, unformatted form, and processes and their resulting formats radiate from it. More practically, though, a web-shaped workflow uses the approach many websites do—that of applying formatting to existing content, rendered in the browser, each time this content is accessed (rather than storing static pages), and the tools (content management systems and blogging platforms like WordPress, Drupal, Joomla, Bricolage, EZPublish, and others) that make building websites more accessible to those with limited resources can be applied to this problem. Publishers can create a web-shaped workflow by creating a *web-first workflow* (see Maxwell and Fraser 2010). This approach is detailed in the next chapter.

3. Web-first at Arsenal Pulp Press

IN THE SUMMER of 2010, this report's author attempted to implement this web-first system at Arsenal Pulp Press. WordPress, a free, open-source blogging and content management platform, was used successfully to create the *Book of MPub*, so Arsenal set out to host, edit, and structurally mark up a trade book, *Farewell My Concubine*, in the same system. This section details the processes of setting up the site, editing in WordPress, exporting the book for various formats, and producing these final formats. It also considers some additional and alternative tools.

3.1 Setting up

ARSENAL PULP PRESS was already using WordPress as an online content management system prior to this project's inception. Richard Swain, the Arsenal webmaster, used WordPress's multi-site function to create an independent website as part of the arsenalpulp.com domain to host *Farewell My Concubine*. The Arsenal staff was already familiar with WordPress as a content management system, so little training was necessary.

The website had to function as more than a content management system, however. To take advantage of the markup capabilities of TinyMCE (WordPress's native text editor) and WordPress's separation of content from format, editors must work directly in WordPress. The book as hosted at editorial.arsenalpulp.com would ideally always be the canonical version of the text, making the workflow system efficient and accessible. In setting up the site, therefore, our primary concerns were both its functionality as a site for exporting content and its usability as an editorial tool. These considerations were informed by *The Book of MPub*, the netCase Editorial Workflow System,[12] and workshopping with Arsenal Pulp's associate editor, Susan Safyan.

12 A research project conducted by six members of the 2009–2010 SFU Master of Publishing cohort attempting to create an online workflow system for a literary magazine, comprising a submission utility and a private review environment. Hereinafter abbreviated as nEWS (Everton et al. 2010).

In selecting a site theme,[13] the team first thought to use one with paragraph-level comment functions to aid the developmental editing process (for examples of this model, see the web versions of Christopher Kelty's *Two Bits* and Peter Salus's *The Daemon, the GNU, and the Penguin*). Arsenal first considered digress.it, which provides paragraph- and page-level commenting, but ultimately discarded it because it did not tolerate sidebars, which were necessary for an export button; without this, extracting the post's XHTML would be laborious. The team experimented with CommentPress, which offers the paragraph-by-paragraph comment function of digress.it, as well as a more robust structure, including sidebars and other structural accessories, but ultimately eliminated it as well, for two reasons. First, XHTML files exported from the site included the comment boxes, and the existing style sheet for converting to InDesign's markup language would have to be altered to accommodate this. Second, and more problematic, the Arsenal editing strategy does not include peer review, and the stages of editing are made less discrete by the small staff. The editor of a particular book is often responsible for substantive as well as sentence-level edits, and these are sometimes presented to the author at the same stage. Therefore, developmental editing and direct engagement with the text at the level of the line or word would be undertaken by the same person, sometimes at the same time, and the latter would have to be performed from the back end[14] of the website. The team soon realized the editor would rarely need or want to view the published version of the post, and instead would need to comment directly in the post body, from the back end. Once it was determined there was no need for the theme to allow the editor to engage meaningfully with the text from the front end, Canvas, a theme that makes CSS easily customizable, was selected.

With the theme in place, the team installed a series of plugins. Much of this process is indebted to nEWS: several plugins were essential to creating a multi-user editing environment.

• After the Deadline. This is an advanced spell-check tool that uses context to determine misspellings, and also detects clichés, passive voice, and other stylistic errors.

---

13 That is, a package of files supporting the structure and appearance of the site.

14 The interface a website's administrators accesses to edit the site's content and appearance. The back end is not public-facing.

• Capability Manager. This plugin assigns roles and permissions to site users. It keeps content private, restricts administrative functions, and streamlines the dashboard an administrator sees when accessing the site's back end.

• Show Post Busy Status. WordPress can save changes from only one user at a time. This plugin prevents users from working on the same document at the same time.

• WP-CMS Post Control. WordPress has a built-in autosave function. Although this prevents data loss, it also vastly increases the number of saved revisions, and the saving process is disruptive in that it reloads the page and scrolls the text up to the beginning, causing the editor to lose her place. The WP-CMS plugin allows users to disable autosave and otherwise customize the functions of the site's back end.

Arsenal also installed two plugins that are a necessary part of a web-first workflow.

• ickmull. This plugin was developed in the Master of Publishing program to export a page's post content as clean XHTML. It appears as a link in the blog's sidebar. This export can then be imported directly into an EPUB file or transformed into ICML for InDesign.

• TinyMCE Advanced. TinyMCE is the visual editor native to WordPress. It uses simple XHTML to mark up content and consistently produces markup that conforms to the syntactical constraints of XML rather than the more lax constraints of HTML, and therefore acts as the filter that prepares content for clean export. TinyMCE Advanced increases both the functionality and customizability of the visual editor; in particular, it allows the user to create custom CSS classes, so complex semantic markup can be performed with the visual editor's drop-down Styles menu.

Finally, the team selected settings that made the book's content private. This included blocking search engines, publishing posts privately, and making user registration by invitation only. Were this a public-facing site, the company would have developed the blog's appearance; because the site was not being used as the book's or publisher's online presence, and because all the editing took place in the site's back end, Arsenal was concerned mostly with the administrative functions and less with the front-end appearance.

as the team at Arsenal experimented with the site's settings, the editor and the intern tested the editing functions with some dummy content and several approaches to editing. First, the team used the site's built-in comment functions to make substantive suggestions, but this proved inefficient because the editor had to move between windows (blog post and back end) to edit at different levels, as would the author when reviewing suggestions and making proposed changes. Next, the editor made changes directly in the post and compared revisions using the built-in version control; the team hoped this function would provide accountability, as versions were labelled by username, and also preserve the original content as Microsoft Word does with its track changes function. Although the compare revisions function met these requirements, the display was difficult to read, as the type was small and the text was the marked-up HTML version, and users still had to switch between windows in this solution. Finally, the team settled on using the inserted and deleted text buttons in TinyMCE's array, making changes directly in the post by selecting text to delete and marking it accordingly, and entering new text and marking it as inserted. The team also created a CSS class for queries, so they could be entered directly into the post and remain clearly demarcated. This imitated the familiar interface of Microsoft Word, without requiring its complex markup: the tags for inserted and deleted text are the standard <ins> and <del>, respectively. This also allowed users to see changes while working in the visual editor, instead of requiring them to navigate between windows. However, the system is less efficient and more error-prone than Word's Track Changes, as users have to mark content as inserted or deleted instead of having their changes detected by the CMS; it also does not support differentiation among multiple users, in that their changes are all marked the same way. Ultimately, Arsenal decided this was the best solution available at the time. Arsenal created a set of instructions called "Working with WordPress" and sent them to the author of *Farewell My Concubine*, Helen Hok-Sze Leung.

Helen had received the standard Arsenal Pulp Press guide to preparing manuscripts for editing, so she submitted the manuscript as a Word document. It was then pasted into a series of blog posts, giving each structural element (chapters

and front and back matter) its own post. WordPress has a built-in filter for content from Microsoft Word, eliminating inline formatting and generating simple semantic markup; this means content can move from MS Word to WordPress at essentially any point, and the editor will not need to intervene into the code.

The book's editor made her first pass through the manuscript using the Word-like system described above, and Susan then reviewed the changes. Team members began to initial queries so the source was clear. Once this round of edits was finalized, Helen was alerted by email and she reviewed the changes. She was able to use the editing system without any difficulty; Arsenal is already accustomed to training authors in using MS Word, so WordPress will probably not be difficult to adjust to in terms of training.

The major downside to editing in WordPress became apparent when Helen had approved the changes and it came time to accept them. There was no straightforward way to eliminate "deleted" text and remove insertion tags. Ultimately, the team worked through the posts manually, removing content marked for deletion, then copied the HTML into Notepad++, a free plain-text editor for PCs, and used the search and replace function to remove insertion tags and any stray deletion tags. Finally, this text was pasted back into the HTML view of the post.

This clean version of the manuscript then went to Brian Lam, the publisher. He elected not to work in WordPress; instead, he printed off the blog posts and worked on paper. Because the design process had not focused on the site appearance, formatting for elements like block quotes and headings did not follow Arsenal style; it was unclear from the team's instructions that the way the content appeared online was not the way it would appear in layout, so the paper manuscript came back with extensive formatting corrections. If Arsenal staff wishes to continue to work on paper in this manner, the CSS for the front end should be edited to avoid formatting confusion.

3.3 Production

THE TEAM BEGAN to export files from the blog using the ickmull button, saving them all to a single folder. The style sheet counterpart to the ickmull plugin was used to

convert these files to ICML. The XSLT (extensible style sheet language transformations) file converts XHTML markup to ICML (InCopy Markup Language), which is the branch of InDesign Markup Language that deals with textual content. These ICML files can then be placed into InDesign, and this content remains linked to the source file so subsequent changes to these files are preserved in the InDesign document.

XSL transformations can be performed from the command line, but Notepad++ and its XML Tools plugin were used for this project. This program allows the user to specify the XSLT file and transform the XHTML files in a simple interface, and then save the resulting files. This XSLT file converts the styles described in the XHTML or XML file into a new format—in this case, into IDML, so they can be read by InDesign.

Once the files exported from the site were transformed, the new files were placed into InDesign. *Farewell My Concubine* is part of Arsenal's Queer Film Classics series, so the text design was already established. The paragraph and character styles were imported from existing Queer Film Classics files and applied to the imported content.

This layout process began before editing was complete, so as changes were made to the text, the InDesign file was updated. The benefits of the web-first workflow system apply only if the website remains the canonical version of the text, and the final formats exist as the result of processes applied to this discrete text, so editing continued in WordPress. Because the ICML files act in place of InDesign's InCopy system, which allows users to edit the text outside the layout, when they are updated or replaced, these changes are applied also to the InDesign document. Edited content was re-exported, converted to ICML, and used to replaced the existing ICML files. As editing drew to a close, however, the WordPress and InDesign files began to diverge. *Farewell*, like the other Queer Film Classics, includes images from the film, and the canonical web version of the text included image numbers and captions; these had to be stripped out of the main text and placed with their corresponding images, and once this process began, the files could not be updated to reflect changes in WordPress. The few changes that occurred after this point had to be inputted in both the web and print-layout versions. In future, this report's author would recommend the image captions and placement information exist in a separate file. Layout of images and descriptive information for them cannot yet be automated, but web-first users can at least minimize disruption of other automated processes.

The layout for print was straightforward, and there were no errors in the converted text. WordPress stripped out Microsoft Word formatting completely, and TinyMCE's markup combined with InDesign's styles, augmented with GREP functions,[15] allowed the team to automate text formatting entirely.

3.4 Launch of Anthologize

the anthologize plugin appeared as a new alternative for producing web-first books in multiple formats as this project was underway. Anthologize, developed by One Week | One Tool at the Center for History and New Media at George Mason University, is a WordPress plugin that promises to export blog content in several formats, including EPUB (Casden et al. 2010). For publishers managing content in WordPress, a direct-to-EPUB function is ideal. The team installed the plugin and experimented with using it in the context of this workflow.

Many of the problems with this plugin will probably be solved as it moves out of alpha, but some are simply a result of the developers' intent to make it a general-purpose export tool for converting public web content into different formats. The Anthologize team will probably rectify its inability to handle images and occasional failed exports. The developer's intentions are more problematic. First, Anthologize cannot find private posts, so book content had to be copied and pasted into the "Import" interface. Second, Anthologize caters to content that exists on the web at large, not to content generated directly in WordPress, to allow users to import RSS feeds from anywhere and export them as ebooks or print-ready formats. As a result, it assumes lowest-common-denominator encoding, not the UTF-8 character set used by WordPress; curly quotes and em dashes were garbled, and the solutions to this problem are inefficient enough to make dedicated software for creating and editing EPUB files manually a more useful and viable option than Anthologize. Finally, issues of limited customizability ruled out Anthologize entirely: Arsenal has ebook standards that include a copyright page and a hyperlinked table of contents, which users cannot achieve in Anthologize, and minor formatting customization is

15 Global search and replace functions that can be built into paragraph styles in InDesign.

also an important part of creating an attractive, useful ebook. Ultimately, although Anthologize will be useful to individual users and self-published authors, the team decided to package XHTML files exported with the ickmull plugin as EPUB files themselves.

3.5 Creating EPUBs in Sigil

sigil is a free, open-source EPUB editor (Markovic 2011). Users can import HTML files directly, or paste content into the WYSIWYG editor. The team imported the XHTML files exported from the website; Sigil automatically generates a style sheet, manifest, and table of contents, and zips the document. Importing HTML files into Sigil creates a viable ebook, but images must be inserted manually (this is easily accomplished in the Sigil WYSIWYG editor). In addition, some minor (hand-hewn) design and copyfitting measures can make the text more readable.

The starting point for using Sigil was uploading content to WordPress in several files, divided into chapters. This was partially for pragmatic reasons—longer files sometimes caused errors when loading or saving in WordPress. More importantly, in order to create a linked table of contents and a useable spine in the EPUB, the book would have had to be divided into separate files; file creation is easier in WordPress than in Sigil.

In creating the *Farewell* ebook, the Arsenal team found that appropriate formatting practices emerged. One of the myriad benefits of ebooks is the flexibility offered to readers. Not only can users manipulate the type, including size, font, whitespace, etc., but they can also intervene with the experience of reading—accessible ebooks allow visually impaired readers to navigate them aurally, for instance. So it makes little sense to take a dictatorial approach to CSS by determining type characteristics, page breaks, and so on. Information architecture becomes more important, and the ebook designer must instead maximize the flexibility of the text while retaining the features that make it intelligible; i.e. the structural elements like captions and block quotes must be semantically marked, and the aesthetic cues must be proportional rather than absolute. The first part of this task is neatly accomplished with WordPress-native XHTML. In the case of *Farewell*, CSS was used to define marked

elements in relation to default text to accomplish aesthetic differences, giving the font size of captions as 0.9 em of the default text, and marking block quotes with a left indent. As for images, it seemed easiest to define the image width as 100 percent relative to the display, allowing the images to resize for different screens. However, due to file size constraints, it might have been wiser to simply use small, anchored images, especially in an image-heavy book like this one. The images in *Farewell My Concubine*, while many, are supplemental rather than central to the text; in some cases large files will be unavoidable, which will present problems of either bandwidth or storage (or both). The problem of images in ebooks could be a report unto itself.

Another aspect of this flexibility is that line and page breaks are unpredictable. This lifts some of the copyfitting burden from the shoulders of the designer, but in *Farewell* some words broke in undesirable ways. For instance, one word with a parenthetical prefix would break between parenthesis and root without a hyphen. Those words that needed to remain unbroken were manually marked and then designated non-breaking spans in the CSS.

Some further measures could have been taken to improve navigability. The Arsenal team created a hyperlinked table of contents for *Farewell*, and footnotes could also have been linked. Linking capability gives the publisher an opportunity to change the way peripheral elements, such as footnotes, glossaries, appendices, etc., are incorporated into a book, and there is a great deal of room here for publishers to experiment with layout and consider the (interactive) reading experience.

Periodically, the team proofed the ebook for formatting errors. Initially the text was proofed with several different ereading programs, but some support less CSS than others. As a result, some measures taken to improve the EPUB's design led to errors in programs like Adobe Digital Editions, which supports very limited CSS and few special characters. Ultimately the book was proofed in the least CSS-supportive program throughout the process of building the EPUB, and final checks were performed in several readers.

Finally, the team used Calibre, "a free and open source e-book library management application," to generate a MOBI file[16] from the EPUB (Goyal 2011). At this

16 An ebook format based on the Open Ebook specifications (like EPUB) but with added security. This format is now owned by Amazon. (Mobipocket 2011)

point, the book existed as a private blog, a print-ready digital file, and a downloadable PDF, EPUB, and MOBI. The canonical source file, the blog, was exported into the print production workflow and the digital production workflow concurrently, and in turn, these workflows forked into multiple formats. Crucially, at no point in the production process did a file have to be stripped of excess markup, and each step made use of the semantic markup established in the editorial process.

3.6 Creating a user guide

Throughout the production process, this report's author drafted a basic user guide. This expanded on the "Working with WordPress" document created to train authors and editors, covering the system's setup, the export process, and the creation of EPUBs in addition to editorial practices. The user guide is reprinted in the appendix of this paper.

4. Successes and failures of the project

ULTIMATELY, THE WEB-FIRST workflow resulted in a successful editing and production process as detailed in Section 3. As well, on the personnel side, all members of the team, including the designer/production manager, editor/supervisor, and author, were able to use the system after minor training.

However, editorial functions need improvement—there is some loss of functionality, in particular Track Changes, moving from MS Word to WordPress, and although some users may be comfortable with this, and the approach to editing may change, in the short term it's too big a transition for most editors and authors (see Section 5 for further discussion of editing in an agile environment). Whether these issues are critical or merely inconvenient likely depends on the staff involved; in Arsenal's case, the editor felt the WordPress editing environment did not provide the functionality she required, and many are likely to agree with her.

WordPress is open-source software, and it has a large, well-supported development community. Many of the problems with the existing WordPress-based workflow could realistically be solved by users developing plugins to supplement the software.

In the immediate future, a number of developments could solve the shortcomings of the systems as it stands, including:

A robust track changes function for WordPress. This system would need to support multiple users, and marking and accepting changes would need to be automatic rather than manual.

An improved version control system in the WordPress back end. If the compare revisions function allowed users to review individual changes and provided a more readable display, this might provide enough accountability to make all parties more comfortable with editing in the system without track changes.

Many publishers have an interest in resolving these shortcomings, including companies like Amazon that support self-publishing through online document submission. Independent developers like Anthologize and PressBooks, both developing WordPress-based web-first publishing platforms, will also create and refine resources available to publishers.

In the long term, there are a number of obstacles to making the web-first system supportive for editors. This is an easier system to learn than proprietary XML-specific editors but it lacks some functionality. Editing is a precision job, and it needs precision tools. Those who have an interest in the development of a web-first system need to make WordPress into a precision editing platform, with the ability to provide editors and authors with a comfortable, transparent revision environment before editors and authors will adopt it. At the same time, one must be mindful of Gall's Law, the notion that complex systems must evolve from simple systems to be successful, and not introduce needless or precarious complexities. The practice of editing will have to adapt to digital environments as much as digital environments will have to adapt to editors. This concept is addressed in Section 5.

In this vein, the current approach to editing must give way to a new approach to editing that is both looser (focusing more on the text in its current state than on the edits made by individual editors at a given point) and more transparent (as a CMS like WordPress is able to record and make public all changes, the time they are changed, the user, and so on). See section 5 of this paper for further discussion.

These concerns aside, the web-first workflow developed for The Book of MPub is usable as is and makes ebook production much more efficient than the print-centric workflow. In addition, because WordPress successfully converts content in Microsoft Word to clean XHTML, editors can migrate content to the website after editing is completed; the only stage that must be performed in WordPress is semantic markup. Publishers who wish to take advantage of WordPress's markup capabilities without abandoning a traditional editing model can still adopt the production aspects of the web-first system, integrating existing editorial practices into the new production process according to their needs.

5. Agile editing

AS DETAILED IN section 4, the agile (web-first) production environment can be considered established or at least feasible given the favourable results of the processes described in Section 3. The web-based editorial workflow, however, leaves something to be desired in its failure to support the familiar track changes system developed by Microsoft. What, then, of the role of the editor? This section addresses the role of the editor in a web-first world.

5.1 Authority and integrity

EDITOR JAMES O'SHEA Wade summarizes the relationship between editors and authors as follows:

> While our relationships with authors are formalized in written contracts, the heart of what we do takes place in phone calls and conversations, in the human friction of brief and hasty interchanges. Clarity in communicating in such informal ways can stave off a lot of grief for author and editor. It is desirable to be clear and make sure you are understood—and that you understand. … This is not a question of moral delicacy; clarity and truthfulness make it possible to carry on a business that depends on mutual trust. (James O'Shea Wade, 78).

In 2010, James Bridle published *The Iraq War: A Historiography of Wikipedia Changelogs*. Spanning twelve volumes, the text is a direct transcription of every edit made to the Wikipedia article on the Iraq War between December 2004 and November 2009. In Bridle's words:

> This is historiography. This is what culture actually looks like: a process of argument, of dissenting and accreting opinion, of gradual and not always correct codification.

> And for the first time in history, we're building a system that, perhaps only for a brief time but certainly for the moment, is capable of recording every single one of those infinitely valuable pieces of information. Everything should have a history button. We need to talk about historiography, to surface this process, to challenge absolutist narratives of the past, and thus, those of the present and our future (Bridle 2010b).

Although Wikipedia articles do not wear their edits on their sleeves, highlighting additions and deletions as Microsoft Word's track changes function does, the history of a Wikipedia article is as vital to the text as the current revision is, and it is every bit as accessible. Arguably, in the age of search engine indexing, caching, stored revisions, and the massive and omnipresent digital memory we call the web, this extends to every website, both public and private. For historiographers, it is the discrepancy between revisions that can be most interesting, and the transparency of the web makes this study more viable than ever.

For editors, too, a history of changes is paramount. Editors rely on their chosen technology to provide transparency and accountability. Before the fluidity of digital files, this was not a concern: neither the editor nor the author can intervene seamlessly with a printed manuscript. These are the benefits of digital files: they are flexible, replicable, and transmittable. But while this was a great advantage for the industry at large, it became a source of anxiety for many editors. This anxiety is described in a chapter entitled "Authenticating Electronic Editions" in the 2006 MLA publication *Electronic Textual Editing*:

> A book is generally seen as a trustworthy carrier of text because, once printed, text cannot be changed without leaving obvious physical evidence. This stability is accompanied by a corresponding inflexibility. Apart from handwritten marginal annotation, there is little augmentation or manipulation available to the user of a printed text. Electronic texts are far more malleable. They can be modified with great ease and speed…The nature of the medium makes the potential effect of these modifications greater because the different versions of the text can be quickly duplicated and distributed, beyond recall by the editor. Does the electronic future, then, hold in store something akin to medieval scribal culture? If this lack of control is the

> risk, will scholars be willing to put several years of their lives into the painstaking creation of electronic editions of important historical documents or works of literature and philosophy? (Berrie et al. 2006, 269)

By 2006, the year of this book's publication, certainly most editing was taking place within word-processing software, and practices were already in place to mitigate the perceived dangers of onscreen editing. While many educational and STM publishers like O'Reilly and Wiley use custom proprietary XML-based editing software, many trade publishers use the widely available, affordable and generally accessible Microsoft Word (O'Reilly 2008; Tamblyn 2009). They rely on an agreed set of practices and the transparency of track changes to maintain the "authority and integrity" of a manuscript, in the words of Berrie et al.

But it seems clear that these set practices are the key to this sense of integrity and authority, regardless of the software in which the editing takes place. After all, it is just as easy to sabotage or misplace edits (as seems to be a primary source of the anxiety Berrie et al. discuss) in Microsoft Word's track changes as in any other digital editing environment. Editors and authors must (and do already) enter the editorial process with the understanding that each wants only to improve the book and each will only take actions that are in the text's best interest. As James O'Shea Wade writes, there is only one way to reconcile editors' allegiance to their publishers with their relationship to authors: "The editor's primary obligation is to the *book*. If you fail in that you are no friend to the author and you are not doing what a publisher pays you to do" (74).

It is in this spirit of collaboration and mutual beneficence that Wikipedia thrives. In fact, according to a 2006 study, on average experts ranked Wikipedia articles related to their fields as "credible" and the author of the study suggests only 13 percent of articles have errors (Chesney 2006). Contributors to a free, crowd-sourced public knowledge base lack the professional commitment, risk to reputation, and ties of the editor-author relationship that motivate editors and authors to preserve a text's integrity and still maintain accuracy in 87 percent of articles.

Editors working in slippery digital environments can maintain the integrity of the text by approaching onscreen editing as they have always approached editing—that

is, by communicating expectations clearly, setting out guidelines and practices at the beginning of the editorial process, and, above all, remembering that authors and colleagues are professionals who share their goals for the text.

5.2 The fluid text

literary critic john Bryant defines the fluid text as "any literary work that exists in more than one version…Truth be told, all works—because of the nature of texts and creativity—are fluid texts" (2006, 1). This definition is so broad as to appear useless; for the purposes of this report we must rather consider the transparency of a text, given that all texts can be described as fluid.

But let us determine first what it means for a text to be fluid, using the example of James Bridle's *The Iraq War*. The fluid text is the text that displays some tension between iterations, and to study the fluid text is to consider the significance of that tension. What does it say about the evolution of the text, or of the author's understanding of the subject? Why is this word or passage more apt than that? How does this reflect the author's politics? What assumptions does it make about the reader? All this is to say that the revisions of a text are as integral to the text itself as are its final (that is, most recent) content and form. As Bob Stein wrote of the experience of reading Wikipedia, "I think there is a lot to be learned by studying the points of dissent; indeed the 'truth' is likely to be found in the interstices, where different points of view collide. Network-authored works need to be read in a new way that allows one to focus on the process as well as the end product" (Stein 2006). These points of interest extend even to texts that are the result of collaboration between an author and editor and even those texts that have experienced no external interventions but reflect only the creative processes of a single author.

Considering that all texts can be defined as fluid and that this fluidity is an indispensable part of a text's significance, what then is the role of the editor? If a particular edition of a text can be seen as a cross-section of the vector that is the book, one must accept that a transparent editorial process is in the book's best interest. But the editor's goal remains the same: the improvement of the book to the extent the

editor is capable of, with the understanding that the development of the book is not finite and the process of this development has its own value.

When a text is born digital, it is an opportunity to share this process with a broader audience, whether through multiple ebook releases or by using an editing environment with built-in version control, like a wiki.

5.3 The ephemeral edition

John Oakes and Colin Robinson founded OR Books in 2009 with the intent to embrace the advantages of the internet in their publishing model. According to Oakes, this entails a subscription model whereby books are printed on demand, released in digital formats simultaneously, bundled for discounts, and sold directly to customers from the company's website (Oakes 2010). To make up for the lack of a sales force and the support of Amazon and the brick-and-mortar bookstores, the company attempts to create viral marketing campaigns by producing timely books, using a long lead-time in marketing, and creating book trailers.

Publishers can choose to adopt any or all of these practices, but in particular a model based on printing on demand is the perfect complement to an agile model of editing and production. Agile publishing seeks to minimize the investment of time and resources in books and formats that may be obsolete before they even reach the market, given the fluidity of the text and the rapid interchange of information in the digital age.

POD fulfilment and ebooks both allow the book to remain under construction indefinitely. As a result, a particular edition of the text becomes ephemeral, disposable. The logical extension of the fluid text is the text that shifts from copy to copy. Of course this is extreme, and for reasons of time constraints and the editor's sanity, the revision process of a particular edition of a book should not extend beyond its publication date. But the possibility of error always exists, and the consequences of some errors are great when a publisher has a warehouse full of stock.

Take, for example, the 2010 Penguin Australia title *The Pasta Bible*. Soon after its publication a reader alerted the publisher that the phrase "freshly ground black pepper" had been misprinted as "freshly ground black people." According to the

Guardian, the entire run of 7000 copies was pulped at a cost to the publisher of $20,000 (Lea 2010). It is impossible to say how such an error occurred in the first place, but a person would be hard-pressed to find any book entirely free of typographical errors, not to mention information that is outdated or obsolete. Penguin was able to absorb the blow, but if a small publisher with more limited resources has invested in hundreds or thousands of copies of a book, such an error could be devastating.

Authors, editors, and proofreaders have always been quick to adapt, and the per-purchase fulfillment model OR Books has implemented gives editors the opportunity to reinterpret the life cycle of the text. The strengths and limitations of a web-first book are different from those of a print-first book; though shifting definitions are not new to editors, the web-first environment is. Editors must find a way to adapt the practice of editing to take advantage of these strengths, rather than trying to adapt the environment to fit those practices developed for print-first books. Berrie et al. discuss the scribal period in the context of textual instability and the difficulty of collating variations into a single authoritative text (Berrie et al. 2006, 271). One must accept that in the digital age, the authoritative text is increasingly impractical and undesirable. By using the flexibility digital publishing affords us, we retain the advantages of massive replicability we gained from the printing press while returning to the fluidity of inscription. But this is a practical solution only for the publisher with agile content—content born digital.

Conclusion

AS I BEGAN my time at Arsenal Pulp Press, I intended to set up and test an agile workflow and train staff in its use, adapting the methods developed for *The Book of MPub* to complement a small trade publisher's existing workflow.

In examining the print-first workflow, it becomes clear that the existing process, prioritizing print formats and preserving final, authoritative versions of a text in format-specific files, is inadequate to the demands of digital publishing, especially given the strained staff resources and compressed timelines at many publishers. The agile framework provides a viable alternative to the outdated top-down publishing model currently in place. This framework makes use of readily available technology, much of which is free or relatively low-cost. However, a web-shaped workflow is radically different from the hierarchical print-centric workflow and cannot be implemented incrementally. Rather, it requires a wholesale renovation of the production process, which can be a risky endeavour, not to mention an investment of time few publishers have. Publishers cannot afford to opt out of change, but neither can they afford to sacrifice their routine publishing activities and the limited resources backing them for expensive, uncertain technology. This report seeks to mitigate these risks by providing a reference document describing in detail both the processes and the (low-cost) tools involved in setting up and adapting a web-first workflow.

This paradigm shift is not without implications for the creation and stewardship of the text. As I have outlined above, there exist incompatibilities between existing editorial practices and the environment provided by an online content management system. To resolve these differences, publishers could adapt the workflow to attempt to replicate the advantages of Microsoft Word and to continue to anchor editing practices in the limitations of print, graft the two workflows together where they are most compatible, or embrace a new editorial model.

The editorial climate detailed above is an estimate of how such a model might look. Given editors' ongoing concerns about quality control in the age of electronic

editing, an approach that provides new ways to access and understand texts seems desirable. The advantages of the ebook and the short-run edition make such an approach feasible, as demonstrated by OR Books.

In 2004, *Wired*'s Chris Anderson coined the term *the long tail*. In brief, it is an asymptotic curve wherein a product's sales approach but never reach zero, and over time the number of units described in the tapering tail is greater than the number in the initial sales spike (Anderson 2004). As Anderson explains, as the book market becomes saturated and each book faces increasing competition, such a sales trajectory is only possible in the online world, with the help of aggregators like Amazon improving books' discoverability:

> People are going deep into the catalog, down the long, long list of available titles, far past what's available at Blockbuster Video, Tower Records, and Barnes & Noble. And the more they find, the more they like. As they wander further from the beaten path, they discover their taste is not as mainstream as they thought (or as they had been led to believe by marketing, a lack of alternatives, and a hit-driven culture).

The long tail, to some degree, makes the concepts of frontlist and backlist titles meaningless, because although publishers allocate most of their resources to frontlist titles, the bulk of a book's sales can be expected to occur at long intervals, years into its life.

In a world where the long tail rules, the long print run is obsolete (for the vast majority of titles): though the run may sell through over its lifespan, it is also a risky venture with up-front costs and slow payoff, it will monopolize warehouse real estate, and, in the case of instructional, scientific, or other non-fiction genres that rely on accuracy, it will rapidly become outdated. How can a publisher predict the longevity or popularity of a particular format? Will the publisher survive if it must absorb the costs of an intolerable error? How many titles should a publisher maintain in the warehouse, in what quantities, and for how long? And why should a publisher, pressed for time, accept the risk of error or obsolescence when a text's lifespan is unlimited?

The web-first model detailed in this paper brings digital publishing within reach for many publishers who do not have the resources to purchase or develop digital publishing software that respects existing print-first practices. Although the system has its shortcomings on the level of practicalities, it is also a working structural remodel and is a viable starting point. Tools for the web can easily be supplemented with more familiar tools for print until a more perfect workflow is developed.

In any case, publishers must find a way to weather the uncertainties of the future, from the financial to the technological. They must produce and archive content that will adapt to new formats. They must minimize costs and risks, expand their audiences, and acquire and develop books that can withstand the test of time. The only solution is to opt for an agile model—that is, a model demonstrating "ease and grace of movement" and acuity. A model unencumbered by prescriptive production methods, practices developed for the static nature of print books, and long print runs. A model both pragmatic and optimistic, in which content is born digital and flows naturally into the format that best suits the rhetorical situation. A model that both embraces the changing technological landscape and roots itself in a new tradition of best practices: a web-first model.

Appendix

User Guide for the Arsenal Editorial Workflow System

(Hosted at http://editorial.arsenalpulp.com/category/User-Guide/)

Setting Up

In setting up a WordPress site to act as a content management system, there are several usability issues to consider. The following is a description of the customization of the Arsenal EWS. This is influenced by the editorial workflow system developed by netCase.

Plugins

After the Deadline

An advanced spellchecking tool, After the Deadline highlights misspellings, grammar errors, and stylistic concerns. Although this is more accurate than many comparable tools, it is much slower than the tools built into browsers and is turned off by default. Useful for authors and editors with the time and inclination.

Anthologize

Anthologize was developed during One Week | One Tool at the Center for History and New Media at George Mason University. It exports blog content to EPUB, PDF, RTF, and TEI. Anthologize is, at the time of writing, still in alpha, so the output is less than perfect, but it will improve. Read more about using Anthologize in a web-first workflow system in Making an Ebook.

Capability Manager

This plugin allows site administrators to create user roles and define their capabilities. Authors must only be able to read and edit their own work, while editors must

have access to all content as well as the ability to adjust settings. Currently only two levels (authors and administrators) are in use, but more hierarchical structures are also possible. The plugin allows administrators to create, delete, and rename user roles and control their capabilities on the level of individual tasks.

ickmull

This plugin, developed by John Maxwell et al., exports post content as clean XHTML. Using the tkbr2icml.xslt style sheet, this export can be placed into InDesign. This is described in detail in Print Production.

Show Post Busy Status

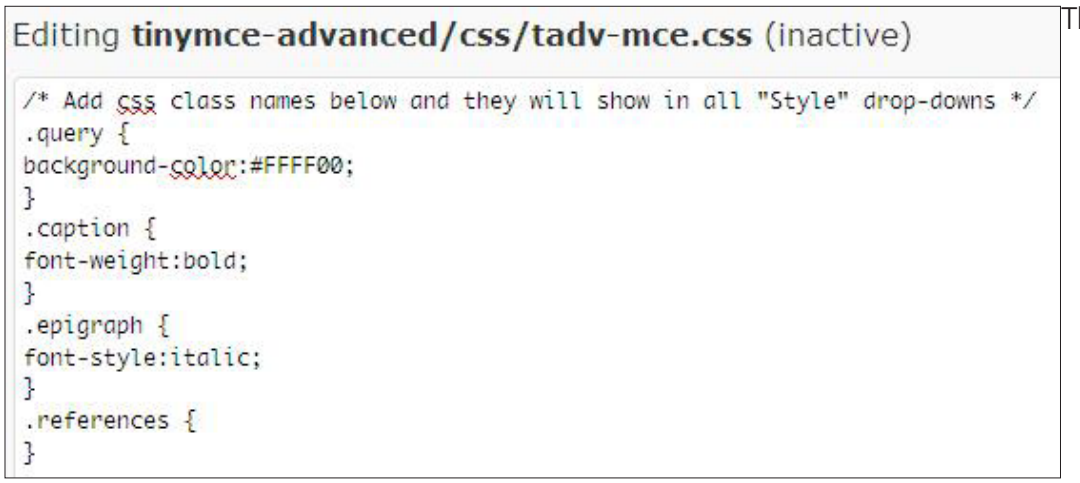
Because WordPress can't save changes made by more than one user at a time, the Show Post Busy Status plugin is a useful tool that helps users avoid losing edits.

TinyMCE Advanced

TinyMCE is the visual editor built into WordPress. The TinyMCE Advanced plugin adds several buttons to the visual editor. In particular, the styles menu and the block quote, insert, and delete buttons are necessary.

To customize the visual editor:

- In the Settings toolbar, select TinyMCE Advanced. Drag buttons from the menu below onto the visual editor array at the top of the page. You can drag these buttons to rearrange them once added to the array.
- Beneath the menu, in a box labelled "Advanced," click on the link to the tadv-mce.css file. This is where to add the paragraph or character styles you wish to use in your InDesign template. Styles you add here will appear in your style drop-down menu.



The

CSS editor for TinyMCE

Styling added here will display in the TinyMCE editor; if you want to change the appearance of text on the front end of the blog, you will need to add custom CSS to your theme as well. This styling will also not carry through to InDesign, but the XHTML tags will, which is all that's needed to use InDesign paragraph and character styles; in fact, if you list classes here with no changes to the styling, that will suffice (see .references in this figure). A useful CSS resource is W3Schools. For the purposes of customizing the Styles dropdown, however, you can use the model at left. Select a name for your class, and insert ".classname { }" on a new line. Save your changes, and the class will appear in the Styles menu in the visual editor.

WP-CMS Post Control

This plugin allows you to customize the post and page controls for user levels, and also allows control of some core functions. This plugin has been used to disable autosave to control the number of revisions stored and to eliminate the disruptive saving process; disabling autosave can make the site more usable but authors and editors need to be more vigilant about saving changes.

Miscellaneous tweaks

Theme options

The Arsenal Pulp Press EWS currently uses Canvas, but any theme that tolerates sidebars and is reasonably customizable would work. On the Widgets page, add the ickmull plugin to a sidebar so content can be exported, and add any custom CSS you wish to use to your theme. Customize the appearance and navigation of the site as best suits your users.

Privacy

Book content on the Arsenal EWS site is private. On the Privacy settings page, search engines are blocked, although all other visitors can reach the site. Within each post, users can select a post visibility: Public, Password protected (designate a shared password with which anyone can reach the post), and Private (allow only logged-in users to read the post). Making posts private eliminates the difficulty of sharing a specific password with the users that need to access the site, and access to private posts can

also be restricted using Capability Manager while password-protected posts cannot.

Settings

In General settings, to limit site users to those approved by administrators, de-select "Anyone can register."

On the Writing settings page, it is useful to de-select "Convert emoticons" and select "Correct invalidly nested XHTML."

Experiment with other settings to improve usability and efficiency.

Working with WordPress

Logging In

Go to http://editorial.arsenalpulp.com, click on "Log in" in the right-hand sidebar, and enter your username and password, or go to http://editorial.arsenalpulp.com/wp-admin/ and enter your username and password.

Uploading Content

Once you have logged in and reached your dashboard, go into "Posts" (in the left-hand navigation bar) and click "Add new."

Title the post with the chapter number and title.

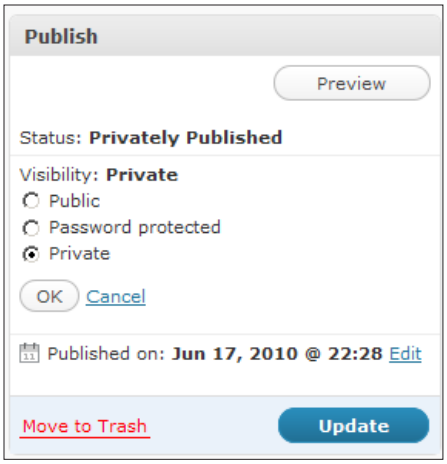
If you are pasting content from Microsoft Word or other word processing software, click the button that shows a clipboard and a W. If you are pasting plain text generated in a text editor (for example, Notepad or TextEdit), click the button that shows a clipboard and a T.

Paste the contents of the chapter into the dialogue box that appears and click "Insert."

In the "Publish" toolbar in the upper right-hand corner, select "Private" visibility.

Click "Publish."

Repeat for all chapters.



As long as your work is "Privately Published," it is secure. Private is preferable to Password Protected because private posts can only be found if you have a login (visitors who go to the URL of a private post see a 404 error message) and they don't require a shared password—only your individual username and password.

Using the "WYSIWYG" Editor

WordPress offers two options for editing content: "What you see is what you get" and HTML. The WYSIWYG ("wizzy-wig") or Visual editor offers many of the same functions as word-processing software like MS Word. Please use the WYSIWYG editor to make your changes, and please don't experiment with font, colour, size, etc.

If you hover your cursor over any of the buttons in the WYSIWYG editor, the button's title will pop up. This will also show you a keyboard command you can use if you prefer hotkeys to mouse navigation. You will never need to use most of these buttons for the Arsenal EWS. The most important buttons are shown below.

Row 1, left to right: Make text italic; bulleted list; numbered list; WordPress spell check; find a word/phrase.
Row 2: Paste from Word; insert a symbol; highlight; help.
Row 3: Delete this text; insert this text.

Making Changes

To edit a chapter, go into "Posts" and select the chapter you wish to edit.

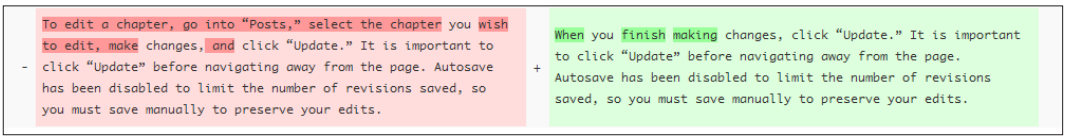
If you want to delete text, highlight the text and click on the button that shows an A struck through in red. If you want to insert text, click the button that shows an A underlined in green.

When you finish making changes, click “Update.” It is important to click “Update” before navigating away from the page. Autosave has been disabled to limit the number of revisions saved, so you must save manually to preserve your edits.

Anything you add to the text will now show up as underlined, while anything you delete will have a strikethrough. You can see these changes in the editor, in the blog version (click on “view post”), and in compared revisions (see below).

Note: If the “Posts” toolbar indicates that the post is busy, do not enter the post. WordPress can save only one user’s changes at a time, and the other user’s changes will be lost.

Reviewing Changes



Both in the editor and on the blog, deleted text will be struck through and inserted text will be inserted. If you wish to review changes once they have been accepted, or if you wish to compare or restore older versions, on the “Edit Post” page, scroll to the bottom and click on one of the revisions (make sure you have saved any changes). Select the revisions you wish to compare and click “Compare Revisions.” The versions will be laid out side by side, and differences will be highlighted—red for dead copy, green for the most recent version.

Queries

If you have a query, please insert it directly into the text set off with square brackets [like this], select it, and, using the dropdown Styles menu, mark it as a query.

Spell Check

There are three options for checking your spelling in the Arsenal EWS.

- WordPress's default spellchecker. Click the button that shows "ABC" and a blue check mark. Words that are misspelled will be underlined. Click one to see suggested spellings.
- The "After the Deadline" spellcheck. To use this, click the button showing "ABC" and a green check mark. This button will underline misspelled or misused words and even trite phrasing. Click an underlined phrase to view suggestions. You can go into your profile to configure what the spellchecker looks for: select any phrasing you want AtD to warn you about. Although this tool isn't perfect, it is more sophisticated than many spelling/grammar checkers available. However, it doesn't offer dictionaries in Canadian or British dialect. You can add words or phrases to the "ignore" list by clicking on them when AtD has underlined them.
- Use your browser's built-in spellchecker. Your browser's "Help" function can help you configure this, and many browsers offer Canadian English dictionaries.

Print Prodution

Exporting content

The ickmull plugin makes it easy to export content. In any post, simply click "Save as Clean XHTML" and the file will download to your computer as export.HTML. Although you can download all the posts on a category page or a blog front page as a single file, you won't be able to change the order of the posts. The best course of action is probably to download each chapter separately by clicking through to each post, and then to use careful file-naming practices and save all exports to a single folder.

Converting to ICML/IDML

Adobe uses ICML (InCopy Markup Language) and IDML (InDesign Markup Language) to mark up its content and apply styles to it; WordPress and many websites use XHTML (Extensible Hypertext Markup Language). Once blog contents have been exported as XHTML, Extensible Style sheet Language Transformations are

used to convert them to ICML—this is easier than it sounds. John Maxwell et al. have created a style sheet which is available for download here. A plain-text editor with XML-editing capabilities can be used to run the transformations. On a PC, you can use Notepad++ with the XML Tools plugin. On a Mac, TextWrangler with XSLPalette works well.

Notepad++ transformations

1. Open one of the exported files in Notepad++.
2. Go to Plugins > XML Tools > XSL Transformation.
3. Select the style sheet file and hit "Transform." This will generate a new file.
4. Save the new file with the extension .icml.
5. Repeat for all the contents of the book.

XSLPalette transformations

1. Open XSLPalette. Select an export as the source file (you can drag the file into the box or browse) and specify the style sheet as the style file.
2. Click "Transform." Copy the output text.
3. Open TextWrangler and paste in the copied text.
4. Save with the extension .icml.
5. Repeat for all contents of the book.

Moving into InDesign

1. Open a new InDesign file according to the book's specifications.
2. Place the first file into the document, holding the shift key to autoflow.
3. Format the book by changing the definitions of paragraph and character styles. InDesign creates a style for each tag used in the source file.
4. If changes are made to the content on the web, export and convert that file again and replace the original .icml file with the new one. In the InDesign document, the links panel will show an error; click the link to update it, and the changes will appear in the InDesign document.
5. To make changes to the contents in the InDesign document, right click on the text and select InCopy > Check out. When content is checked out and changes are made, you can no longer update from the blog without losing those changes.

Making an Ebook

Making an Ebook with Anthologize

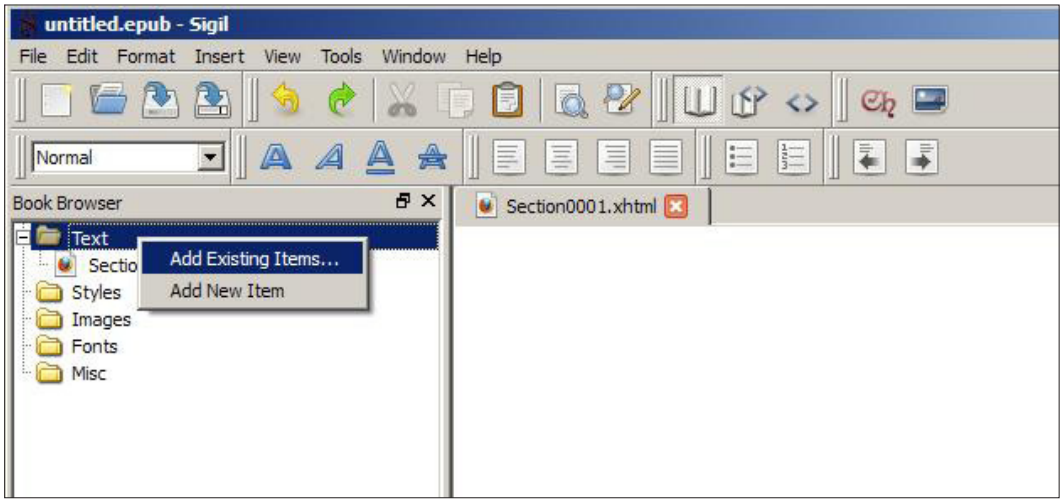
Anthologize is a new WordPress plugin developed at the Center for History and New Media. It is a tool that allows users to export the content of WordPress blogs in several formats: EPUB, PDF, RTF, TEI. This tool is easy to use but, in its current release, does not accommodate Unicode characters like em dashes and curly quotes, and image support is still buggy.

1. Select "New Project," fill in title and author information, and save.
2. Click on "Manage Parts." From here, you can create new parts and drag and drop content into them. Note: Anthologize requires that all files be assigned to parts.
3. Anthologize cannot find private posts (although it can find unpublished drafts), so if the contents of your book are marked private you will now have to import content. Select "Add New" under "Imported Items" in the sidebar. This will bring you to the standard WordPress editor, where you can paste in content and format it as you normally would. Note that content you save here is visible only to site administrators. Imported content is visible on the "manage parts" page and you can now drag it into a part.
4. When you are ready, click on "Export Project." You will be asked to enter limited metadata (author, copyright information, dedication, and acknowledgments). You can then select the output file format, page size, font style, and font size.
5. Clicking "Export" will open your downloads window, where you will find your finished EPUB.

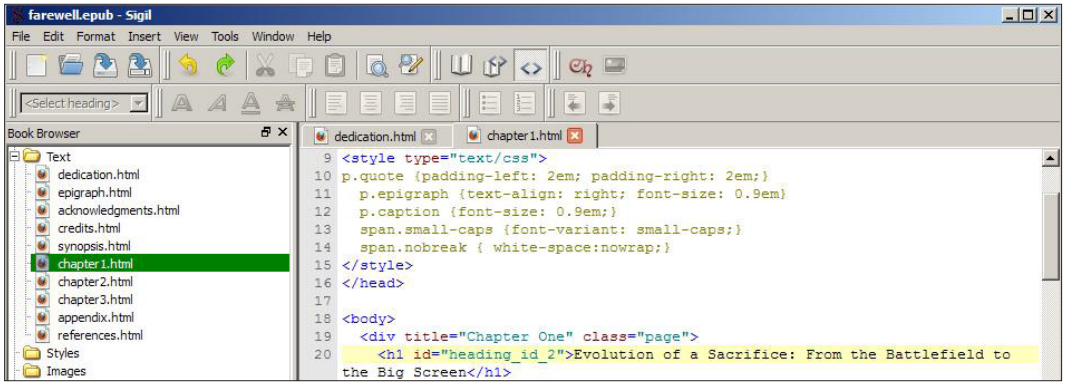
Making an Ebook from Clean XHTML Exports

An EPUB file is simply bundled XHTML files, a style sheet, and some self-generated elements like a table of contents. Content exported from WordPress blogs can be bundled into an ebook in an ebook editing program. These instructions are for Sigil, an open-source, "multi-platform WYSIWYG ebook editor." Other options include the free (but not open-source) eCub, which does not allow WYSIWYG editing, and proprietary programs like Adobe Dreamweaver.

1. In WordPress, on a post page, select "Export as Clean XHTML" (note that this is enabled by the ickmull plugin designed by John Maxwell, Meghan Macdonald, et al.). This will download the post content to your computer as a file with an HTML extension.
2. In Sigil, right-click on the "Text" folder in the left-hand sidebar and select "Add Existing Items." Select all the (HTML) files you want to include in the ebook. These items will appear in the sidebar, where you can also remove the default blank file Sigil has created and drag the files into the order in which you want them to appear.



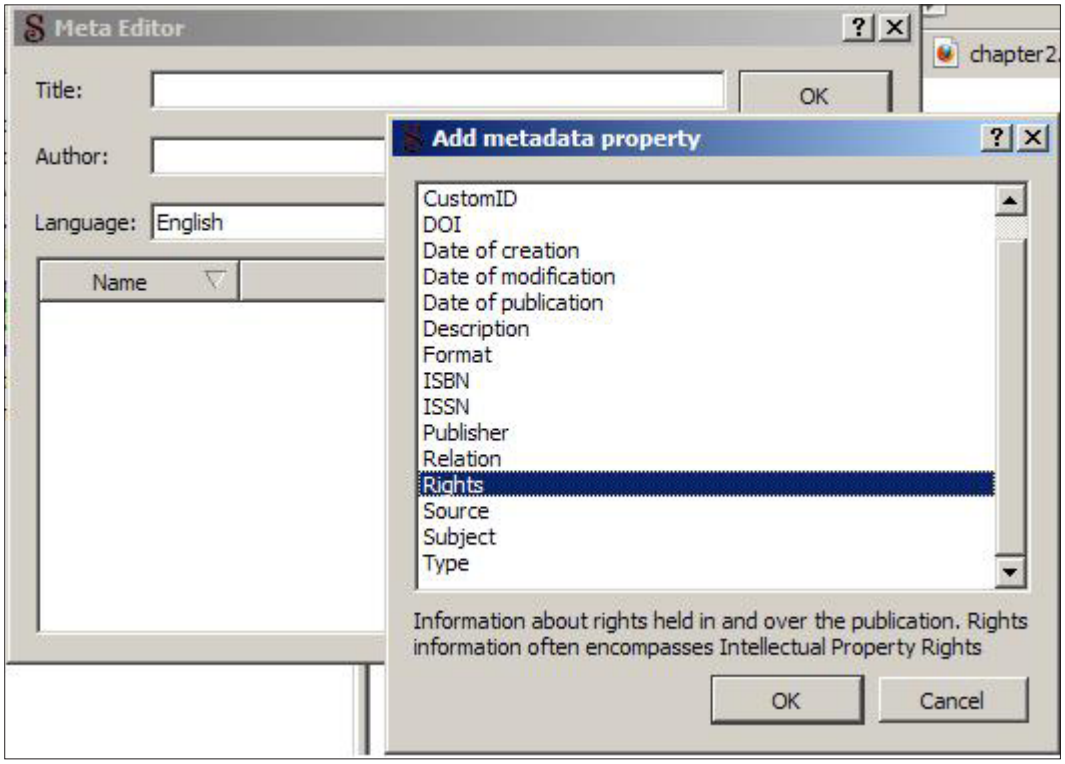
3. Click on one of the files to edit it. It will automatically open in Book View, which is the WYSIWYG editor. From here, you can edit the formatting but not customize the cascading style sheets (CSS). Sigil recognizes standard tags like h1, p, and quote, and will format them accordingly, but to accommodate custom tags, like epigraph or caption, you will have to either format each instance separately in the visual editor or customize the internal CSS using the Split or Code View.



4. In this figure you can see custom internal CSS written to accommodate epigraphs, captions, small caps, and sequences that should not be broken at the end of a line. This belongs in the document head (that is, it should appear between <head> and </head> in the Split or Code View). Note that internal CSS should appear between <style type="text/css"> and </style>. In the body of the document, these sequences are tagged accordingly. Note that these tags must be nested (that is, tags can appear inside other tags but cannot overlap them; in this figure, the "small caps" span begins and ends within the "caption" tags. The sequence Figure… is ill-formed).

```
27
28      <p class="caption"><span class="small-caps">Figure</span> 1 DVD still.
Dieyi asks: "Why does Concubine Yu have to die?" Chen Kaige, <em>Farewell My
Concubine</em>, 1993.</p>
```

5. Metadata is an important part of making an ebook discoverable. Ebook editing software (like Sigil) allows you to select the fields that apply to your book and fill them out.



6. Save the file and it's done.

Making an Ebook from Scratch

If you are making an ebook from files in Word, much of the process is the same as above; however, it will require more hand-coding.

1. As above, right-click on the "text" folder in the left-hand sidebar. Select "Add New Item." Here, you can paste in content from Word or other word-processing software; most of the formatting will be lost.
2. Basic formatting (bolding, italics, indents, alignment, and lists) can be performed in the WYSIWYG editor, or you can use the Split or Code View to create custom CSS and tag content accordingly (as above). Again, Sigil will recognize common tags like h1 through h6 for headings, p for paragraphs, quote, em or i for italic, and strong or b for bold.
3. Add metadata and save as above.

Tips and Resources

- EPUB files are compressed the same way .zip files are. To crack open any EPUB, simply rename the file so the extension is .zip instead of .EPUB and unzip the file.
- EPUB files can be proofread in Adobe Digital Editions or online readers like Ibis on a computer; on devices, in addition to proprietary software specific to the manufacturer, you can use Ibis, Stanza, Kobo, or other applications. It is ideal to check the file in as many places as possible to make sure the text reflows properly and all your formatting is supported. For example, Adobe Digital Editions does not support small caps, but Ibis does; books proofed in Ibis may be error-laden when read in Digital Editions.
- To learn more about CSS and XHTML, visit W3Schools.
-

References

Adobe. 2011. Adobe InDesign 5.5. http://www.adobe.com/products/indesign.HTML (accessed May 18, 2011).

———. 2011. Adobe PageMaker 7. http://www.adobe.com/products/pagemaker/ (accessed May 18, 2011).

———. 2010. IDML file format specification version 7.0. Adobe Systems Incorporated.

"agile, adj.". *OED Online*. Oxford University Press. http://www.oed.com.proxy.lib.sfu.ca/view/Entry/3979?redirectedFrom=agile (accessed May 18, 2011).

Anderson, C. 2004. The long tail. *Wired*, October. http://www.wired.com/wired/archive/12.10/tail.html?pg=1&topic=tail&topic_set= (accessed May 18, 2011).

Arsenal Pulp Press (Arsenal). 2010. Arsenal Author Kit. Internal document.

Beck, K., M. Beedle, A. van Bennekum, A. Cockburn, W. Cunningham, M. Fowler, J. Grenning et al. 2001. Manifesto for agile software development. http://agile-manifesto.org/ (accessed May 18, 2011).

Berrie, P., P. Eggert, C. Tiffin, and G. Barwell. 2006. Authenticating electronic editions. In *Electronic textual editing*, ed. L. Burnard, K.O. O'Keeffe, and J. Unsworth, 269–276. New York: The Modern Language Association of America.

Bridle, J. 2010a. *The Iraq war: A historiography of Wikipedia changelogs*. Self-published.

———. 2010b. On Wikipedia, cultural patrimony, and historiography. http://book-two.org/notebook/wikipedia-historiography/ (accessed May 18, 2011).

Bryant, J. 2002. *The fluid text: A theory of revision and editing for book and screen*. Ann Arbor: The University of Michigan Press.

Casden, J., B. Gorges, K. Gossett, S. Hanrath, E. Kapsalis, D. Knox, Z. McCune, et al. 2010. Anthologize. http://anthologize.org/ (accessed May 18, 2011).

Chan, V., C. Ferguson, K. Fraser, C. Geissler, A. Metten, and S. Smith. 2010. *The book of MPub: New perspectives on technology and publishing*. Vancouver:

Canadian Centre for Studies in Publishing & Pressplay.

Chesney, T. 2006. An empirical examination of Wikipedia's credibility. *First Monday* 11.11. http://firstmonday.org/htbin/cgiwrap/bin/ojs/index.php/fm/article/view/1413/1331 (accessed May 18, 2011).

Everton, K., K. Hilderman, L. Kemp, E. Quintana, S. Smart, and C. Theriault. 2010. *netCase editorial workflow system*. Simon Fraser University Master of Publishing report.

Fraser, K. 2010. XML: More like a unicorn than the iPad, but still not really a unicorn. In *The book of MPub*, ed. V. Chan et al. Vancouver: Canadian Centre for Studies in Publishing & Pressplay.

Gall, J. 1978. *SYSTEMANTICS: How systems really work and how they fail*. Pocket.

Goldberg, K.H. 2008. *XML: Visual QuickStart guide, second edition*. Berkely: Peachpit Press. http://proquest.safaribooksonline.com.proxy.lib.sfu.ca/book/xml/9780321602589 (accessed May 18, 2011).

Goyal, Kovid. 2011. Calibre. http://calibre-ebook.com/ (accessed May 18, 2011).

Hauben, R. 2004. The internet: on its international origins and collaborative vision. *Amateur Computerist* 12 (Spring). http://www.ais.org/~jrh/acn/ACn12-2.a03.txt (accessed May 18, 2011).

Ho, D. 2011. Notepad++. http://notepad-plus-plus.org/ (accessed May 18, 2011).

Hunt, A. and D. Thomas. 2011. Pragmatic Programmers: About. http://pragprog.com/about (accessed May 18, 2011).

International Digital Publishing Forum (IDPF). 2011. *International digital publishing forum: Trade and standards organization for the digital publishing industry*, s.v. "EPUB." http://idpf.org/EPUB (accessed May 18, 2011).

International Paper Company (IPC). 1934, 2007. *Pocket pal: The handy book of graphic arts production*. Memphis: International Paper Company.

Kasdorf, W.E. 2003. *The Columbia guide to digital publishing*. New York: Columbia University Press.

Kelty, Christopher. 2008. *Two bits: The cultural significance of free software*. Durham and London: Duke University Press. http://twobits.net/discuss/ (accessed May 18, 2011).

Kist, Joost. 2009. *New thinking for 21st-century publishers: Emerging patterns and evolving stratagems*. Oxford: Chandos Publishing.

Lea, R. 2010. Penguin cookbook calls for "freshly ground black people." *The Guardian*, April 19. http://www.guardian.co.uk/books/2010/apr/19/penguin-cook-book (accessed May 18, 2011).

Leung, H.H. 2010. *Farewell my concubine*. Vancouver: Arsenal Pulp Press.

Markovic, S. 2011. Sigil. http://code.google.com/p/sigil/ (accessed May 18, 2011).

Maxwell, J. and K. Fraser. 2010. Traversing *The book of MPub*: An agile, web-first publishing model. *The Journal of Electronic Publishing* 13.3. http://quod.lib.umich.edu/cgi/t/text/text-idx?c=jep;view=text;rgn=main;id no=3336451.0013.303 (accessed May 18, 2011).

Mobipocket. 2011. Mobipocket: About. http://www.mobipocket.com/en/Corporate/AboutMobipocket.asp?Language=EN (accessed May 18, 2011).

Oakes, J. 2010. Book Publishers Should Embrace the Digital Age. *London Progressive Journal*, April 16. http://londonprogressivejournal.com/article/676/book-publishers-should-embrace-the-digital-age (accessed May 18, 2011).

O'Reilly XML Blog. 2008. http://www.oreillynet.com/xml/blog/ (accessed May 18, 2011).

Quark. 2011. QuarkXPress: Xperience design. http://www.quark.com/Products/QuarkXPress/ (accessed May 18, 2011)

Rankin, M. Tools of Change round-up day 3. InDesignSecrets.com. http://indesignsecrets.com/tools-of-change-round-up-day-3.php (accessed July 16, 2011).

Rosenblatt, R. 1983. A new world dawns. *Time*, January 3. http://www.time.com/time/magazine/article/0,9171,953631,00.HTML (accessed May 18, 2011).

Salus, P. 2008. *The daemon, the GNU, and the penguin*. Groklaw. http://www.groklaw.net/staticpages/index.php?page=20051013231901859 (accessed May 18, 2011).

Stein, B. 2006. Jaron Lanier's Essay on "The Hazards of the New Online Collectivism." *If:book: An institute for the future of the book*. http://www.futureofthebook.org/blog/archives/2006/08/jaron_laniers_essay_on_digital.HTML (accessed May 18, 2011).

Steinberg, S.H. 1955, 1961, 1995. *Five hundred years of printing*. The British

Library & Oak Knoll Press.

Tamblyn, Michael. 2009. An XML publishing workflow that doesn't suck. BookNet Canada Tech Forum 2009. http://www.slideshare.net/booknetcanada/bnctechforummichaeltamblyn (accessed May 18, 2011).

W3C. 1999. *HTML 4.01 specification*. http://www.w3.org/TR/html4/ (accessed May 18, 2011).

W3C. 2002. *XHTML 1.0: The extensible HyperText markup language (second edition)*. http://www.w3.org/TR/xhtml1/ (accessed May 18, 2011).

Wade, J.O. 1993. Doing good—and doing it right: The ethical and moral dimensions of editing. In *Editors on editing: What writers need to know about what editors do*, ed. G. Gross, 73–82. New York: Grove Press.

Young, S. 2007. The book is dead: long live the book. Sydney: University of New South Wales Press.