Electronic Publishing: Politics and Pragmatics

New Technologies in Medieval and Renaissance Studies
Volume 2

ITER

Medieval and Renaissance Texts and Studies
Volume 401

Electronic Publishing: Politics and Pragmatics

Edited by
Gabriel Egan, Loughborough University

NTMRS Series Editors
William R. Bowen and Raymond G. Siemens

Iter Inc.
in collaboration with
ACMRS
(Arizona Center for Medieval and Renaissance Studies)
Tempe, Arizona
2010

The publication of this volume has been supported by grants
from the Senate of Victoria University in the University of Toronto
and the Renaissance Society of America.

Published by Iter, Inc. and
ACMRS (Arizona Center for Medieval and Renaissance Studies)
Tempe, Arizona

Library of Congress Cataloging-in-Publication Data

Electronic publishing : politics and pragmatics / edited by Gabriel Egan.
p. cm. -- (New technologies in medieval and Renaissance studies ; v. 2)
(Medieval and Renaissance texts and studies ; v. 401)
Includes bibliographical references.
ISBN 978-0-86698-449-2 (alk. paper)
1. Scholarly electronic publishing. 2. Communication in learning and
scholarship--Technological innovations. 3. Humanities--Digital libraries.
4. Library materials--Digitization. 5. Criticism, Textual--Data processing.
6. Transmission of texts. I. Egan, Gabriel.
Z286.E43E4385 2010
070.5'797--dc22

2010043523

ISBN 978–0–86698–021–0 (online)

∞
This book is made to last.
It was typeset in SIL Gentium at the
Institute for Research in Classical Philosophy and Science (Princeton).
It is smyth-sewn and printed on acid-free paper to library specifications.
Printed in the United States of America

Contents

Editor's Acknowledgements ix

Introduction
Gabriel Egan 1

Part One. Creating Electronic Publications: The Politics and Pragmatics of Tools, Standards, and Skills 15

The Impact of Computers on the Art of Scholarly Editing
Peter Shillingsburg 17

Digitizing George Herbert's *Temple*
Robert Whalen 31

A First-Principles Reinvention of Software Tools for Creative Writing and Text Analysis in the Twenty-First Century
Jeff Smith 63

Mechanick Exercises: The Question of Technical Competence in Digital Scholarly Editing
Alan Galey 81

A Historical Intermezzo: Is TEI the Right Way? 103

SGML, Interpretation, and the Two Muses: A Critique of TEI P3 from the End of the Century
Ian Lancashire 105

Lancashire's Two Muses: A Belated Reply
Murray McGillivray 121

Part Two. Disseminating Electronic Publications: The Politics and Pragmatics of Publication 137

How We Been Publishing the Wrong Way, and How We Might Publish a Better Way
Peter Robinson 139

Open Access and Digital Libraries: A Case Study of the Text Creation Partnership
Shawn Martin 157

From Edition to Experience: Feeling the Way towards User-Focussed Interfaces
Paul Vetch 171

The Book of English: Towards Digital Intertextuality and a Second-Generation Digital Library
Martin Mueller 185

Afterword
John Lavagnino 205

Contributors 215

Mechanick Exercises: The Question of Technical Competence in Digital Scholarly Editing

Alan Galey
University of Toronto
alan.galey@utoronto.ca

Technical competence and the unknown

> There are known knowns; there are things we know that we know. There are known unknowns; that is to say, there are things we *now* know we don't know. But there are also unknown unknowns; there are things we *do not* know we don't know.
>
> Donald Rumsfeld, U.S. Defense Department Briefing, 12 February 2002

To borrow the former U.S. Defense Secretary's infamous epistemology, how might we characterize the various types of unknowns in digital scholarly editing, and what do editors need to know about using the Web as a delivery platform? Ever a cause for anxiety and wistfulness among overextended scholars, this kind of question takes on a particular urgency as born-digital scholarly editions become viable. (By *born digital* I mean not simply editions that begin life as computer files—nearly all scholarly writing must be born digital in this sense by now—but rather editions designed for use primarily in digital environments rather than print.) The traditional humanities curriculum can equip new scholars with knowledge of bibliography (enumerative, analytical, historical, descriptive, and textual [Greetham 1994, 5–8]), palaeography and codicology, history of the book trades, stemmatics, languages, literary criticism, editorial theory, biography, intellectual history, reception history, theatre history, archival research methods, and other core competencies; but what does digital editing add to this already formidable list? A recent job advertisement for a "Postdoctoral Fellow in Early Modern Textual Studies and Digital Humanities" invites applicants with competence in "TEI P5; XML, XSLT, XSL and XHTML encoding; XQuery; eXist XML databases; JavaScript; Ruby on Rails; PHP; CSS; and web-based SQL database projects using PostgresSQL and mySQL" (ETCL 2008).

ISBN 978–0–86698–021–0 (online) ISBN 978–0–86698–449–2 (print)
New Technologies in Medieval and Renaissance Studies 2 (2010) 81–101

Such piling-up of knowledge domains on an editor's list of things to learn can seem as daunting as the labours of Hercules, the greatest mythic to-do list of antiquity. Erasmus even compared it to the thankless labours of textual scholarship in his *Adages*: "I should like to know who would not be frightened off ... from engaging in such work, unless he be a real Hercules in mind, able to do and suffer anything for the sake of serving others" (Erasmus 1964, 194; Jardine 1993, 41–45). Erasmus lived in a time of scholar-printers such as John Froben and Aldus Manutius, and their legacy of encyclopaedic skill sets is still detectable in Joseph Moxon's account of the printer's art from 1683–84 (our earliest surviving manual of the trade):

> A *Corrector* should (besides the *English* Tongue) be well skilled in Languages, especially in those that are used to be Printed with us, *viz*. the *Latin*, *Greek*, *Hebrew*, *Syriack*, *Caldæ*, *French*, *Spanish*, *Italian*, *High Dutch*, *Saxon*, *Low Dutch*, *Welch*, *&c*. neither ought my innumerating only these be a stint to his skill in the number of them. (Moxon 1683, 2:260, Mm4v)

Erasmus's question remained a good one in Moxon's time, and even five centuries later. Continuing the trope of the copious list, Jerome McGann responds to digital competence-inflation in the persona of a besieged humanities scholar: "What are you saying? Learn UNIX, hypermedia design, one or more programming languages, or textual markup and its discontents? Learn bibliography and the sociology of texts, ancient and modern textual theory, history of the book?" (McGann 2005, 107). McGann's answer, and the one reflected in the state of the field today, is a clear yes: these are exactly what textual scholars and other humanists must learn if they are to have a voice in the digital reconstitution of the human record.

Yet we lack a name for the type of individual who embodies this synthesis within scholarly editing. The term *corpus editor*, as defined by Gregory Crane, David Bamman, and Alison Jones, describes something close, in that he or she "occupies a middle ground between the algorithm-heavy, knowledge-light approaches of computer science and the wholly manual practices of traditional editing" (Crane, Bamman, and Jones 2007, 52), but their definition still depends upon Fordist notions of specialization. Many computer scientists have always been knowledge-heavy—the best ones, in my experience, make a virtue of curiosity that puts humanists to shame—and many traditional editors have always combined "manual practices" with theoretical inquiry. Frederick Brooks, in his landmark book on the organization of labour and knowledge in software engineering, had to reach outside his own discipline

for the term *architect* as a metaphor for the individual entrusted with the conceptual integrity of a project (Brooks 1995, 41–50 & passim).

It was the need to move beyond this two-cultures divide which prompted Northrop Frye, in an unlikely but resonant keynote address to the 1989 conference of the Association for Computing in the Humanities, to remind us that

> three of the most seminal mechanical inventions ever devised, the alphabet, the printing press, and the book, have been in humanist hands for centuries. The prestige of humanists in the past came largely from the fact that they lived in a far more efficient technological world than most of their contemporaries. It is true that today they are sometimes confused about the new possibilities opening up in front of them, though hardly more so than the rest of the human race. (Frye 1989, 8)

Although scholarly editors have been looking to digital technologists for answers to questions about the relation of labour to knowledge, it is worth heeding Frye's reminder that humanists have themselves been technologists and information architects all along.

If the emerging field of digital textual studies lacks a clear answer to my initial question—what does a digital scholarly editor need to know?—it is because any answer depends upon complex relationships between labour, epistemology, and technology, which extend beyond any primarily technical discussion. The following chapter applies the Rumsfeldian taxonomy to scholarly editing and suggests ways digital editors can address questions of technical competence based on the close historical parallels between the digital humanities and textual studies as fields. For the purposes of this discussion, I take *digital editor* to mean anyone undertaking a scholarly edition or similar project designed for delivery over the Web (at this time the main delivery system for digital editions), and *technical competence* to mean the minimum knowledge required for progress in the absence of specialist contractors and research assistants.

Known knowns: Humanities computing and textual scholarship

In the inaugural volume of *Literary and Linguistic Computing* Susan Hockey reports on a workshop held at Vassar College in 1986 that addressed the same question of what humanists need to know about computing (Hockey 1986, 228). She remarks that the place of programming in the curriculum generat-

ed much discussion but little consensus apart from agreement that the now-outdated languages "PASCAL, BASIC, and SNOBOL were all thought suitable" for humanists to learn at the amateur level (Hockey 1986, 228). However, the rationales posited by workshop members at the time have endured: that programming inculcates "mental discipline" just as Latin language training has done for centuries; that programming reveals computers' capabilities and limitations alike; and that programming stretches the mind to inspire new ways of thinking about problems (Hockey 1986, 228). More recently, Geoffrey Rockwell echoes the same desire for high-level integration: "The important thing is the integration of skills preparation with intellectual preparation. We shouldn't hide skills and technique—they are what makes [sic] digital humanities different from other programs. Instead, we should think of our programs as an art" (Rockwell 2003, 243).

The integration Hockey and Rockwell describe has long been an ideal difficult to achieve in practice. Peter Shillingsburg similarly attempts to integrate the skill sets of editing and computing, though his book's most recent edition was published before the emergence of the Web as the dominant delivery system (Shillingsburg 1996). Typesetting and other document-centric ways of conceptualizing digital editions are inadequate to the mixed ontologies of Web 2.0, where it is becoming increasingly difficult to tell the difference between a page and a program. Attempts to articulate stable known knowns for digital editing illustrate the value of understanding programming's benefits in abstract terms, as an intellectual exercise independent of any particular language or tool.

The editors of a recent Modern Language Association (MLA) collection, *Electronic Textual Editing*, note the problem that "there are currently few manuals, summer courses, or self-guided tutorials that would help even trained textual editors transfer their skills from print to electronic works" (Burnard, O'Keeffe, and Unsworth 2006, 16). (An exception is University of Victoria's Digital Humanities Summer Institute, which combines the best aspects of a skills workshop, international conference, and summer camp.) However, Burnard and his co-authors write from amid the abbreviations that represent scholarly editing in its most institutionalized form: the MLA's Committee on Scholarly Editions (CSE), formerly the Centre for Editions of American Authors (CEAA), and the Text Encoding Initiative (TEI). David Greetham calls this "an *ex cathedra* statement from the policing organization of our discipline(s)" (Greetham 2007, 133). As the publishers of guidelines for scholarly work, the MLA, CSE, and TEI represent textual scholarship's archival tradition: the part of the discipline that shares many librarians', archivists',

and information technologists' reverence for professional standards, best practices, and, most of all, the institutionalizing of a professional desire for "reproducing and uniting the best standards so far developed" (Burnard, O'Keeffe, and Unsworth 2006, 16).

Yet editorial theory and practice over the past quarter-century have come to embody a deep scepticism of institutionalized orthodoxies. Textual scholars today have been trained amid the debates sparked by intensely contested editions such as the Hans Walter Gabler *Ulysses*, the George Kane and E. Talbot Donaldson *Piers Plowman*, the Edinburgh Editions of Walter Scott's *Waverley* novels, the 1986 Oxford *William Shakespeare: The Complete Works* (including the two-text *King Lear*, and a three-text *Lear* in its reincarnation as the Norton Shakespeare), the three-text Arden 3 *Hamlet*, and the 2007 Oxford *Thomas Middleton: The Collected Works*. To these projects we can add more general topics of debate, especially focussed in Shakespeare studies, such as unediting, the (allegedly) bad quartos, multiple-text editions, performance, canonicity, the gendering of texts, the intentions of Shakespeare and his company with regard to publishing playbooks, and the very idea of literature as a distinct category of texts. Should the skills training that comes from "manuals, summer courses, [and] self-guided tutorials" communicate the questions at stake in these debates or simply compartmentalize them?

Few of textual scholarship's known knowns have gone uncontested, especially toward the end of the twentieth century, and there has been little consensus-based progress toward so-called best practices. Instead, dissensus has given us something better: a critical tradition to parallel the archival tradition, advancing knowledge through contestation, debate, and the inclusion of voices from outside the specialist field of scholarly editing, especially from postcolonial studies, book history, performance studies, gender and sexuality studies, film and media studies, and critical theory. To distil editing into a set of institutionally sanctioned practices is to neglect this critical tradition, and to underestimate the challenges it raises. Understanding the often-unacknowledged friction between the critical and archival traditions must be the first thing a digital scholarly editor needs to know, since it will determine how one relates to everything else. This conclusion hardly dispels the unease voiced by Erasmus and McGann, nor does it yet tell us what a digital scholarly editor needs to know. What, then, are the undiscovered regions of the knowledge domain of digital scholarly editing, and how might we identify them on the map of what we do and do not know?

Known unknowns: Foundational technical skills for digital editors

This section describes four technical subjects that are normally the domain of Web programmers, but which digital scholarly editors would do well to understand. I mostly omit discussion of specific software packages because editing is not a task for any one piece of software in the way that a single word processor serves most scholars for writing. I echo William Turkel and Alan MacEachern's premise in their website *The Programming Historian* (required reading for digital editors) favouring a polyglot approach to technologies and languages. No single technology can do everything, despite proprietary software often being marketed commercially as a capital-S Solution (such as Adobe Flash). Encoding, text analysis, and interface design are intellectual endeavours not reducible to abbreviations—text encoding, for example, is about much more than learning any single markup language—but there is nonetheless a need for stable, extensible technologies that enable a range of practice, from the simple to the complex. Some favour the development of a tool or suite of tools, but I am sceptical that these could be anything other than a scholar-built counterpart to Dreamweaver, a tool so generalized that one must bypass its own interface (using the code view pane) to do much specialized or original work. The following list therefore names four areas of knowledge of technical fundamentals that will likely remain pertinent despite the coming and going of software and tools.

1) The history of Web browsers. Browsers, like word processors, are ubiquitous and, therefore, often unnoticed. In the 1990s it was not uncommon to hear the words Netscape or Explorer used to describe any browser regardless of whether it was actually Netscape's Navigator software or Microsoft's rival Internet Explorer. A major recent development in browsers is the maturation of the Web as a platform for delivering applications, not just documents as in the original conception for the Web (Berners-Lee 1990). Scholarly editing is conceptually document-oriented (suited to text encoding) rather than algorithmic (suited to data structures and object-oriented design). Yet browsers are spaces where documents, data structures, and algorithms merge into hybrid forms with complex ontologies. The browser's helpful View Source function gives easy access to a document's source code in HyperText Markup Language (HTML) or eXtensible Markup Language (XML), but increasing complexity in Web architectures means that there is often more to a website than is visible in the source code. Browsers also have a history of conflict with one another and with the very principle of standardized design. Browsers from different makers are not the same, and a single browser may not always function identically on different operating systems.

Ideally one should be able to design Web-based materials according to the independent recommendations of the World Wide Web Consortium (W3C) and have that design function identically on different browsers, as opposed to writing individual workarounds for a specific browser. That ideal was far from reality during the period known as the browser wars, waged largely between Netscape and Internet Explorer since the mid-1990s. We are closer to that ideal now with the Mozilla Firefox browser, which is reasonably standards-compliant and platform-consistent, and to a lesser extent with Safari and recent versions of Internet Explorer, though there are many browsers beyond these major ones, and almost all cling to proprietary implementations of certain features. Wikipedia's entries on Web browsers and their history are useful entry points to this topic (and are reliable at the time of writing), but digital editors should also read a detailed history of this most important of tools (Haigh 2008, 125–47). Digital editors should also become familiar with the online documentation for the major browsers and their design communities, the Mozilla Developer Center and the Microsoft Developer Network.

2) Text encoding and markup. Markup of written texts precedes digital computing by centuries, from word-separation by spacing to modern punctuation (Parkes 1992). Although encoding and markup are sometimes conflated under the term *tagging*, it is necessary to understand the distinction between text encoding formats such as ASCII and Unicode and markup languages such as XML. There is also value in understanding markup in more abstract terms than the adding of tags to texts, but that topic is beyond the scope of this discussion (McGann 2004). In particular, one should be aware of how Unicode translates symbols that stand for letters or punctuation visible on the screen into machine-readable codes and back again (Wittern 2007). This knowledge is important for all editors since the careless migration of digital texts between operating systems and applications can introduce errors even though no one has made a typographical error. The Anglocentric history of computing means that keyboards and software handle accented letters awkwardly, so editors working in languages other than English, especially those using non-Latin alphabets, will need to know Unicode well. Even editors working solely in English may need to account for ligatures, swash letters, and other typographical and scribal phenomena.

Systems for marking up texts with tags have a long and varied history, from typesetting software to Standard Generalized Markup Language (SGML) to its Web-oriented derivative, but it is fair to say that XML has superseded them for the present. Along with XML we have several related technologies for key

tasks: eXtensible Stylesheet Language Transformations (XSLT), for transforming XML into new forms; XPath, for accessing specific sets of entities within XML documents; and XQuery, for running more complex queries than XPath normally handles. These technologies are usually combined with one another; for example, an XSLT stylesheet is itself an XML file, and uses XPath to select parts of the target document to transform. Perhaps the most fundamental value of XML is its capacity to enforce internal consistency and external conformance by means of a schema or Document Type Definition (DTD), a formal abstraction of the rules that the document is supposed to follow.

A project-wide schema or DTD is a tremendously helpful error-control mechanism, and writing it can be an exercise in formalizing one's assumptions about one's material. The TEI guidelines represent a large-scale, multidisciplinary, and collaborative effort to extend this thinking to the full range of materials that humanists might represent using markup. TEI is more than just a tagset; it is also a mechanism for validating files tagged according to its vocabulary, a protocol for customizing its own tagset, a community for sharing tools and resources, and, most valuably, a locus of debate about humanities materials that makes a virtue of the constraints of markup. In other words, the TEI guidelines should be regarded not simply as a solution to a problem, but as a vocabulary to enrich our questions.

The TEI's *Gentle Introduction to XML* is a good place to start, though humanists will need to look further for introductions to XSLT, XPath, and other related technologies (see the TEI Consortium's website for links to resources). Now that the three major browsers contain XML parsers, one can do a great deal even with the simplest combination of tools: a text editor and Web browser. However, it is helpful to have a single XML editing program that allows one to validate files against schemas and DTDs, run XPath queries, execute XSL transformations, and perform other tasks in one place. Currently the proprietary XML editor *oXygen* provides this functionality and includes the TEI schema.

3) Regular expressions. The simplest and most robust tool for searching and manipulating strings of texts is the standardized grammar known as regular expressions. Basic parts of this grammar will be familiar to anyone who has used a wildcard character in a simple search, as when using *Plo*man* to find matches with *Ploughman* or *Plowman*, and *Le?r* to find matches with *Lear* or *Leir*. Since the grammar is standardized, its rules apply more or less consistently across different programming languages and environments, from JavaScript to Perl to PHP (a recursive abbreviation for "PHP: Hypertext Pre-

processor"). Understanding regular expressions has intellectual value beyond practicality; it also disciplines the mind to think about patterns within written language, as Hockey describes, and thus could serve as the basis for user-driven searches of digital editions (Egan 2005).

4) Ajax. Scholars planning digital editions often lack a framework to hold these and other technologies together. With the advent of Firefox as the first popular, standards-compliant, cross-platform Web browser, there has been renewed interest in Web applications that meet third-party standards and build on the free and open-source software ethos. One framework has come to be known by the acronym *Ajax*, or Asynchronous JavaScript and XML, and is a composite of five elements:

> 1. standards-based presentation using eXtensible HTML (XHTML) and Cascading Style Sheets (CSS);
> 2. dynamic display and interaction using the Document Object Model [DOM];
> 3. data interchange and manipulation using XML and [...] XSLT;
> 4. asynchronous data retrieval using the XMLHttpRequest object;
> 5. and JavaScript binding everything together.
>
> (Garrett)

Some implementations of Ajax add two more components: an XML database such as eXist or Tamino, and a server-side language such as PHP or Java to handle interaction between the database and the client-side JavaScript. Further layers may come into play at the server side, usually requiring advanced expertise and access.

In the Ajax architecture, the interface and the underlying functionality that manipulates the data are closely integrated, written in the same language (JavaScript) and located in the same files. Matthew Kirschenbaum suggests why this kind of integration is important: "from a developer's perspective, the interface is often not only conceptually distinct, but also *computationally* distinct" in many older architectures, and "it wasn't until the comparatively recent advent of languages like Visual Basic that it even became practical to program both a user interface and an application's underlying functionality with the same code" (Kirschenbaum 2004, 524–25). This combining of interface and functionality replicates the humanistic design principle that presentational forms are inseparable from analytical functions because a text's meaning is constituted in part by its material form.

Three general points may be added. Firstly, the most important skill, underwriting all others, is the ability to learn new skills quickly and independently. Yet autodidacticism must be balanced with the second point, that learning how to write code does not necessarily mean one has learned to write good code that is concise, legible, efficient, and elegant. Programming is no less an art than rhetoric, and learning the vocabulary of any one language is only the beginning; more advanced topics include data types, data structures, search and sort algorithms, and object-oriented design. Finally, anyone embarking on a digital edition that requires server resources beyond static web page hosting, such as XML databases and server-side scripting in a language like PHP, will soon encounter the always complex interface between the technological and the institutional. Although it is not difficult to set up a local server on one's own computer—using a package such as MAMP (Mac, Apache, MySQL, PHP) for the Apple Macintosh, or its Microsoft Windows counterpart XAMPP (X-platform, Apache, MySQL, PHP, Perl)—doing the same on a university Web server may require considerable skill in navigating the politics and pragmatics of institutional research support.

Unknown unknowns: Interface and divisions of knowledge

Rumsfeld identified the category of the "unknown unknown" as the most dangerous of all, and in digital editing that category has often manifested itself in matters of interface. Many scholarly editing projects begun in the 1990s did not anticipate how difficult and resource-intensive interface design would be (Kirschenbaum 2004, 524–25), nor how many opportunities for new research it presented. Bethany Nowviskie, for example, reflects on the project history of the *Rossetti Archive*:

> Why did we neglect interface until it was almost too late? ... [P]art of the reason ... was ... an unsavory combination of condescension toward and blind faith in programmers. Our own clarity on the subject has increased, I think, precisely in step with our education in technical issues, programming and stylesheet languages—all the things that academics are too likely to leave to others. (Nowviskie 2000)

The lesson here is that project organization and "education in technical issues" must be closely integrated from the outset, not left to "others," lest unknown unknowns manifest themselves without warning.

Projects today still encounter a lack of tools for visualizing the digital edition and making latent intellectual structures meaningful on the screen. This

gap receives little attention because of the wealth of resources researchers now enjoy in related areas like text encoding and text analysis, which have evolved along with their own working languages of choice. A neophyte computing humanist sitting down to learn the fundamentals of text analysis or text encoding soon learns what tools he or she needs on the workbench in order to make something happen and to produce the early results, however modest, that are essential to the success of the autodidactic learning process. This is known as the *hello world!* stage, in which the traditional first task upon learning a new programming language is to make those words appear on the screen. For text analysis and text encoding there are also resources ready to help, like those offered by the TEI and the Text Analysis Portal for Research (TAPoR), but there is no corresponding centralized resource for interface design in the humanities.

One could formulate the problem like an examination question for the present generation of digital humanists:

XML is to markup as Perl is to text analysis

as _____ is to interface design.

One should think carefully before filling in any one acronym or tool. The widespread assumption throughout the late 1990s that hypertext would solve all technical and theoretical problems has proven chimerical. For many theorists and non-specialist commentators, hypertext was an empty vessel into which they poured their hopes without bothering to understand digital texts at the level of code—they simply did not know how much they did not know. The pattern persists, and whereas the technologies mentioned above represent known unknowns, this blank represents the unknown unknown continuing to obtrude into contemporary digital humanities' orderly world of tools.

No single tool or technology can provide the things we have not yet learned to expect from a digital edition, yet consumerist reliance on commercial applications may lead non-specialists to think of interface development as simply a matter of tags, stylesheets, and hyperlinks. The prevailing thinking in software marketing, especially since the late 1990s, crystallizes in the word *solution* as a noun meaning software that solves a problem as soon as it comes out of the box. What You See Is What You Get (WYSIWYG, pronounced whizzywig) Web-design software encapsulates complexity, simplifying design to the point where Web authoring becomes a species of word processing—hence the idea of a web page, symptomatic of the constraints imposed

by document-oriented computing. What you see is what you get, but you get only what you can see. Even Ajax may not be the right answer to the blank in the above homology, since the question may contain a category error. Perhaps we need not so much an acronym as an art, to echo Rockwell.

Among digital editing projects one finds two characteristic responses to the existence of unknown unknowns. One is the DIY ethos in which individual scholars stay on top of the copious desiderata of programming languages and databases. Turkel and MacEachern describe the best rationale for this ethos in *The Programming Historian*: "If you don't program, your research process will always be at the mercy of those who do" (Turkel and MacEachern 2007, ch. 2 sec. 2), which is also the message of the present chapter. However admirable such an approach might be in terms of individual autonomy and work ethic, it is difficult for many scholars to sustain in terms of time and career development. (A helpful document in that regard is an MLA report titled "Guidelines for Evaluating Work with Digital Media in the Modern Languages.") A more debilitating side-effect of prioritizing technical work can be what Anthony Kenny calls "distortion of research" (Kenny 1992, 9), a neglect of the important non-digital aspects of humanities scholarship, especially the critical debates ongoing in one's area of study. As Kenny warns, "There is a danger that projects may be undertaken not because they are likely to lead to academically interesting results, but simply because they are susceptible to computerization" (Kenny 1992, 9).

The other common response to unknown unknowns is the collaborative ethos in which one delegates labour—and with it, knowledge—to team members or contractors, with the lead editor providing skills in project management and funding procurement instead of first-hand technical competence. Shillingsburg argues:

> Scholarly editors are first and foremost textual critics. They are also bibliographers and they know how to conduct literary and historical research. But they are usually not also librarians, typesetters, printers, publishers, book designers, programmers, web-masters, or systems analysts. ... [Thus] textual scholarship requires the services of Internet technologists. (Shillingsburg 2006, 94–96).

Shillingsburg's conclusion, however appealing, perpetuates the black-boxing inherent in the WYSIWYG approach, in which the user can work without ever having to see the underlying code. Similarly, a shared technology services

model like Project Bamboo's, tempting to over-taxed textual scholars, carries the risk that cross-disciplinary influence will flow in only one direction, from the supposedly technology-oriented disciplines like computer science, information studies, and cognitive science to the text-oriented humanities, but not back again. Although encapsulation is essential to computing generally, people are not software, and we should be cautious about applying principles of software design to project organization, interdisciplinarity, and human relationships.

There is a growing division between projects where the humanists regard the technologists as providing services, and those where it is difficult to tell the humanists and the technologists apart. (I now actively avoid the former kind of project.) The division deepens with the belief, popular with digital humanists and funding agencies, that collaboration is self-evidently good. Yet collaborating specialists may dig ever deeper into their disciplinary niches, drawing rigid borders between what they need to know and what they do not. By contrast, it can be liberating and enlightening for a non-technical humanist to start a digital project by painstakingly learning to write her own DTD or schema, treating it as an intellectual exercise in the modelling of materials and not rote implementation of rules dictated by past practice. Ideally, one emerges with a better understanding of one's materials and one's digital tools and of the relationship between them, which in the humanities may be anything but straightforward. The most viable collaborations are intellectual ones, based on the meeting of different minds where all parties emerge changed; the least viable are those that attempt only to balance out skills deficiencies. At worst, the latter approach can render apprenticing scholars (postdoctoral fellows, research assistants) more valuable as continuing subordinates than as future colleagues bound for careers beyond the project.

Thus one may become too technically competent or competent for the wrong reasons. A gifted programmer working on scholarly editions risks exploitative collaborations or conceiving of research materials and questions exclusively in terms of computation. To prevent this, literary and textual scholars must overcome two conceptual constraints. The first is the idea of infrastructure as the guarantor of excellent research. (I use the word *excellent* here mindful of Bill Readings's cautions about its stealth function in academic politics [Readings 1996, 21–43]). Although infrastructure of the kind that Shillingsburg describes is worthwhile and necessary (Shillingsburg 2006, 80–125), the broader institutional discourse still tends to use infrastructure to denote equipment, tools, and shared services rather than knowledge and the capac-

ity to hire knowledgeable people. The second constraint is the assumption of an unbridgeable gap between those working with code and those working with texts and ideas, such that a humanities scholar and a programmer cannot be the same person. To draw a distinction between programming and abstract, poetic thinking is to misunderstand both. Yet in many electronic editing projects the labour divides precisely along these lines: editing stops where interface design begins, with another specialist entering the picture, often as a research assistant rather than an equal collaborator. The division of labour between programming and humanistic inquiry is often necessary, but the division of knowledge is impoverishing.

Conclusion: The missing term

I have argued above that digital editors need to possess technical skills themselves, not just in their research assistants and collaborators, and that the distinction between skills and knowledge is artificial. Does, then, a digital scholarly editor need to be a programmer too or merely able to hire one? If one is committed to the critical tradition of scholarly editing, not just the heaping-up of digitized texts and tools, the answer must be that the project leader should be a programmer, even if she does not do most of the programming. True technical competence cannot be bought. A more heartening rationale for humanists learning programming is described by Stephen Ramsay in a conversation with Turkel and the hosts of the *Digital Campus* podcast, Dan Cohen, Mills Kelly, and Tom Scheinfeldt, in an episode on humanities programming. Ramsay describes the empowerment he felt by learning how to control his digital environment at the level of code—how to "build it, hack it, break the warranty" (Cohen, Mills, and Scheinfeldt 2008, 30.00). For him and his students, upon crossing the programming threshold "the digital world ... suddenly seems ... less deterministic than it might have been before." Kirschenbaum makes a similar case to a mainstream audience in a recent *Chronicle of Higher Education*: "Computers should not be black boxes but rather understood as engines for creating powerful and persuasive models of the world around us. ... I believe that, increasingly, an appreciation of how complex ideas can be imagined and expressed as a set of formal procedures—rules, models, algorithms—in the virtual space of a computer will be an essential element of a humanities education" (Kirschenbaum 2009, B10).

Behind Ramsay's and Kirschenbaum's rationales we may detect the liberal arts tradition of the enlightened, autonomous individual, technically equipped to make informed judgments of a kind that Alan Liu expresses more pointedly: "My highest ambition for cultural criticism and the creative

arts is that they can in tandem become 'ethical hackers' of knowledge work—a problematic role in the information world but one whose general cultural paradigm needs to be explored" (Liu 2004, 7–8). With all these convincing rationales available, what remains to prevent humanists from becoming programmers? All the technological conditions have been in place long before now: computers are cheap; the tools are free (from browsers and text editors to XML databases like eXist, and local server environments like MAMP); online tutorials and knowledge bases are free and ubiquitous; online communities for new programmers are thriving and ready to help (and free); the open-source ethos has resulted in many reusable Web programming components being made available for free; and the Web as a distribution platform is mostly free (though servers and their support do cost money). Where, then, are all the programming humanists?

An explanation for their absence may be found in the fourth, unstated category of Rumsfeld's taxonomy, as Slavoj Žižek explains:

> What [Rumsfeld] forgot to add was the crucial fourth term: "unknown knowns," things we do not know that we know—which is precisely ... the disavowed beliefs and suppositions we are not even aware of adhering to ourselves. ... The situation is like that of the blind spot in our visual field: we do not see the gap, the picture appears continuous. (Žižek 2008, 457)

A disdain for the mechanical constitutes the most potentially troublesome blind spot for digital scholarly editing, and the unknown known it conceals is the idea that computers are venues for labour and not for thinking. This is a distinction as old as the liberal arts that gave rise to the modern university system. By invoking the liberal arts tradition in his defence of learning code, quoted above, Rockwell implicitly contextualizes programming and similar skills within the humanities' known knowns (as Hockey does by invoking Latin). It is worth remembering, though, that the liberal arts tradition draws its rationale from the often-unstated Aristotelian distinction between the liberal arts and the servile arts (Adler 1937, 430–44; Burke 2000, 84). If the liberal arts are traditionally defined as those needed by a free citizen of the state, then the servile (or useful) arts are those needed by the servants of the citizens, like the rude mechanicals of *A Midsummer Night's Dream*.

Yet that dichotomy is transforming into something else even as the difference between digital humanities and other humanities disappears. Liu describes this related ideological formation:

> Wherever the academy looks in the new millennium, it sees the prospect of a world given over to one knowledge—a single, dominant mode of knowledge associated with the information economy and apparently destined to make all other knowledges, especially all historical knowledges, obsolete. Knowledge work harnessed to information technology will now be the sum of all worthwhile knowledge—except, of course, for the knowledge of all the alternative historical modes of knowledge that undergird, overlap with, or—like a shadow world, a shadow web—challenge the conditions of possibility of the millennial new Enlightenment. (Liu 2004, 7)

With Liu's argument in mind, it is worth recalling that the object of Greetham's critique of the MLA's *Electronic Textual Editing* collection was not what knowledge the book offers its readers, but how the book assumes that knowledge should function in the world. If the archival tradition and its conservative worldview embody the kind of illusory continuity that Žižek describes—an ideology, in other words—then we need a critical tradition that operates in the gaps of the digital humanities as an Enlightenment project, making the discontinuities visible.

Such a critical tradition might regard programming as a link to historical modes of knowledge work that resist what Liu calls the "one knowledge" of the information economy. Early modernists in particular should be sensitive to the fluidity between the liberal and servile arts, since the dichotomy began to lose its coherence in the period leading up to Moxon's time in the late seventeenth century (Prest 1987, 13). Indeed, as Jonathan Sawday describes in a chapter on early modern "reasoning engines," Francis Bacon and other seventeenth-century thinkers attempted to rescue the idea of the mechanical from its past associations with socially inferior labour, though at the cost of a certain instrumentalism (Sawday 2007, 210–16). The same ambivalent transformation may happen in our time as digital editing, along with digital knowledge work in the humanities generally, comprise not the "tradesman's entrance" to the academy, as Willard McCarty calls it, but rather "a computing that is *of* as well as *in* the humanities: a continual process of coming to know by manipulating representations" (McCarty 2004, 265). The digital humanities at their best represent not only a synthesis of disciplines, but also a synthesis of different types of labour and knowledge.

Ours is not the first generation of textual scholars to reckon with the problem of mechanical knowledge. Book history, an interdisciplinary cousin and

historical contemporary of the digital humanities, has reckoned with applied technical knowledge in its own conception of its known unknowns. As the New Bibliographers' chronicler F.P. Wilson contends,

> often ... the bibliographer reaches conclusions that are demonstrable and irreversible. The reason is that he is dealing with an Abel Jeffes or a James Roberts not in his relations with other human beings, whether of the government, or the Stationers' Company, or the playhouse, but in his relations with a mechanical process (Wilson 1970, 34).

Bibliography and book history have been negotiating between the different epistemologies of the mechanical, linguistic, and aesthetic for decades. D. F. McKenzie, for example, did not just lead the reintegration of bibliography with book history and literary interpretation, but also established and operated the Wai-te-ata Press at Victoria University of Wellington, New Zealand, using an 1813 Stanhope hand-press that he operated himself and used to teach his students (McKitterick). This figure of the scholar at the press, attempting to guarantee what Brooks called "conceptual integrity" in software design (Brooks 1995, 256), returns us to Moxon's *Mechanick Exercises*:

> *it is necessary that a* Compositor *be a good English Schollar at least; and that he know the present traditional* Spelling *of all English Words, and that he have so much Sence and Reason, as to* Point *his Sentences properly: when to begin a Word with a* Capital Letter, *when (to render the Sence of the Author more intelligent to the Reader) to* Set *some Words or Sentences in* Italick *or* English Letters, *&c.* (Moxon 1683, 2:198, Ee1v)

What Moxon is writing about, and what he demonstrates here in print, is the fundamental link between the details of text and the ubiquity of markup: the latent architecture of information that gets manifested and modified through the productive constraints of a mechanical process. The difference between Moxon's technology and the ones I have described above is one of scale but not of nature. What digital textual scholars need to know, then, may be learned by reckoning with our unknown knowns concerning knowledge work, and by rediscovering what we already know about our own mechanic exercises.

Note

This essay reflects conversations with many people, and I particularly wish to thank Gabriel Egan, Matthew Bouchard, Martin Mueller, Harvey Quamen, Stephen Ramsay, Seamus Ross, students in my 2008–9 graduate seminars, and audiences at gatherings organized by the Society for Digital Humanities, the HCI-Book group, and the Folger Shakespeare Library. Any errors are my own.

WORKS CITED

Adler, Mortimer Jerome. 1937. *Art and Prudence: A Study in Practical Philosophy.* New York: Longmans.

Berners-Lee, Tim. 1990. Information Management: A Proposal. http://www.w3.org/History/1989/proposal.html.

Brooks, Frederick P. 1995. *The Mythical Man-Month: Essays on Software Engineering*. 2nd ed. Reading, MA: Addison Wesley.

Burke, Peter. 2000. *A Social History of Knowledge: From Gutenberg to Diderot.* Cambridge: Polity.

Burnard, Lou, Katherine O'Brien O'Keeffe, and John Unsworth. 2006. Introduction. In *Electronic Textual Editing*, 11–21. New York: Modern Language Association of America.

Cohen, Dan, Kelly Mills, and Tom Scheinfeldt. 2008. Digital Campus Episode 25: Get With the Program. April 21. http://digitalcampus.tv/2008/04/21/episode-25-get-with-the-program/.

Crane, Gregory, David Bamman, and Alison Jones. 2007. ePhilology: When the Books Talk to Their Readers. In *A Companion to Digital Literary Studies*, ed. Susan Schriebmann and Ray Siemens, 29–64. Malden, MA: Blackwell.

Egan, Gabriel. 2005. "Impalpable Hits: Indeterminacy in the Searching of Tagged Shakespearian Texts." A conference paper delivered on 17 March at the 33rd annual meeting of the Shakespeare Association of America in Bermuda, 17–19 March. http://gabrielegan.com/publications/Egan2005a.htm.

Erasmus, Desiderius. 1964. *The "Adages" of Erasmus. A Study with Translations.* Trans. Margaret Mann. Cambridge: Cambridge University Press.

ETCL. 2008. Postdoctoral Fellowship in Early Modern Textual Studies and Digital Humanities (2009–11) at the Electronic Textual Cultures Laboratory (ETCL) at the University of Victoria. *Humanist*. September 11.

Frye, Northrop. 1989. Literary and Mechanical Models. In *Research in Humanities Computing*, ed. Ian Lancashire (Selected Papers from the 1989 Association for Computers and the Humanities-Association for Literary and Linguistic Computing (ACH-ALLC) Conference): 3–12. Vol. 1. Oxford: Clarendon Press.

Garrett, Jesse James. Adaptive Path: Ajax, A New Approach to Web Applications. http://www.adaptivepath.com/ideas/essays/archives/000385.php.

Greetham, D. C. 1994. *Textual Scholarship: An Introduction.* New York: Garland.

_____ 2007. Review of *Electronic Textual Editing*, ed. Lou Burnard, Katherine O'Brien O'Keeffe, and John Unsworth. *Textual Cultures* 2, no. 2: 133–36.

Haigh, Thomas. 2008. Protocols for Profit: Web and E-mail Technologies as Product and Infrastructure. In *The Internet and American Business*, ed. William Aspray and Paul Ceruzzi, 105–58. Cambridge, MA: Massachusetts Institute of Technology Press.

Hockey, Susan. 1986. Workshop on Teaching Computers and the Humanities Courses. *Literary and Linguistic Computing* 1, no. 4: 228–29.

Jardine, Lisa. 1993. *Erasmus, Man of Letters: The Construction of Charisma in Print*. Princeton, NJ: Princeton University Press.

Kenny, Anthony. 1992. *Computers and the Humanities.* London: British Library.

Kirschenbaum, Matthew G. 2004. "So the Colors Cover the Wires": Interface, Aesthetics, and Usability. In *A Companion to Digital Humanities*, ed. Susan Schreibman, Ray Siemens, and John Unsworth, 523–42. Blackwell Companions to Literature and Culture. Oxford: Blackwell.

_____ 2009. Hello Worlds: Why Humanities Students Should Learn to Program. *Chronicle of Higher Education* 55, no. 20 (January 23): B10–B12.

Liu, Alan. 2004. *The Laws of Cool: Knowledge Work and the Culture of Information*. Chicago: University of Chicago Press.

McCarty, Willard. 2004. Modeling: A Study in Words and Meanings. In *A Companion to Digital Humanities*, ed. Susan Schreibman, Ray Siemens, and John Unsworth, 254–70. Blackwell Companions to Literature and Culture. Oxford: Blackwell.

McGann, Jerome J. 2004. Marking Texts of Many Dimensions. In *A Companion to Digital Humanities*, ed. Susan Schriebmann, Ray Siemens, and John Unsworth, 198–217. Blackwell Companions to Literature and Culture. Oxford: Blackwell.

_____ 2005. Information Technology and the Troubled Humanities. *TEXT Technology* 14, no. 2: 105–21.

McKitterick, David. Oxford Dictionary of National Biography article: McKenzie, Donald Francis (1931–99). http://www.oxforddnb.com/view/article/72097.

Moxon, Joseph. 1683. *Mechanick Exercises, or, The Doctrine of Handy Works.* Vol. 2. London: Joseph Moxon.

Nowviskie, Bethany. 2000. 'Interfacing the Rossetti Archive'. Conference paper presented at the October 2000 conference of the Humanities and Technology Association. http://www2.iath.virginia.edu/bpn2f/1866/dgrinterface.html.

Parkes, M. B. 1992. *Pause and Effect: An Introduction to the History of Punctuation in the West*. Aldershot: Scolar.

Prest, Wilfrid. 1987. Introduction: The Professions and Society in Early Modern England. In *The Professions in Early Modern England*, ed. Wilfrid Prest, 1-24. New York: Croom Helm.

Readings, Bill. 1996. *The University in Ruins*. Cambridge MA: Harvard University Press.

Rockwell, Geoffrey. 2003. Graduate Education in Humanities Computing. *Computers and the Humanities* 37, no. 3: 243–44.

Sawday, Jonathan. 2007. *Engines of the Imagination: Renaissance Culture and the Rise of the Machine*. London: Routledge.

Shillingsburg, Peter L. 1996. *Scholarly Editing in the Computer Age.* 3rd ed. Ann Arbor: University of Michigan Press.

_____ 2006. *From Gutenberg to Google.* Cambridge: Cambridge University Press.

Turkel, William J, and Alan MacEachern. 2007. *The Programming Historian.* NiCHE: Network in Canadian History and Environment.

Wilson, F P. 1970. *Shakespeare and the New Bibliography*. Ed. Helen Gardner. Oxford: Clarendon Press.

Wittern, Christian. 2007. Character Encoding. In *A Companion to Digital Literary Studies*, ed. Susan Schreibman and Ray Siemens, 564-576. Malden MA: Blackwell.

Žižek, Slavoj. 2008. *In Defense of Lost Causes*. London: Verso.