Evaluating Open Source Software for Use in Library Initiatives: A Case Study Involving Electronic Publishing

Ruth Gallegos Samuels, Henry Griffy

abstract: This article discusses best practices for evaluating open source software for use in library projects, based on the authors’ experience evaluating electronic publishing solutions. First, it presents a brief review of the literature, emphasizing the need to evaluate open source solutions carefully in order to minimize Total Cost of Ownership. Next, it describes the process used to conduct a comparative evaluation between two open source electronic publishing systems, highlighting aspects of the methodology not described elsewhere. Finally, it concludes with a discussion of reporting the results of an evaluation to decision-makers.

Introduction

The success and viability of digital initiatives hinge on the software chosen to carry out the objectives. It is, therefore, crucial to choose carefully among available alternatives. In this paper, we provide a case-study and commentary about an in-depth evaluation of two open source software solutions for use in the electronic publishing (e-publishing) program at our research university library. We hope that this account will help colleagues who have been tasked with choosing software for their local situation. In addition, some of the methods we developed may prove useful for evaluations of other types of software, especially open source, with potential value for libraries.

Libraries have increasingly taken on initiatives involving complex digital systems, some by necessity – such as those required to provide reliable access to digitized resources...
Evaluating Open Source Software for Use in Library Initiatives

– and some by choice. Several listservs have been instituted to facilitate conversations among librarians about these technologies; and the Association of Research Libraries (ARL) website offers several multi-level sections devoted to policy papers and to support for these initiatives. The trend is clear: libraries are deeply engaged in digital projects and are likely to remain so for the foreseeable future.

These initiatives require a significant investment of time and resources. A recent survey by Tschera Harkness Connell and Thomas Cetwinski, for example, finds that in twenty libraries with Institutional Repositories, an average of 12.4 staff people, constituting an average 4.1 Full time Equivalent (FTE) were involved with that initiative alone. Given the importance and cost of these endeavors, it is urgent that institutions maximize the impact of their efforts, especially as budgets continue to fall short of increased demands. Carefully evaluating the tools used for projects will go a long way toward reducing costs. While this paper focuses specifically on an evaluation of open source electronic publishing platforms, we suspect that our insights are of relevance to projects that require the effective evaluation of other kinds of software tools for scholarly communications.

The State of E-publishing in Academic Libraries

E-publishing has already become a widespread practice among academic libraries. According to a 2007 Association of Research Libraries (ARL) survey,

By late 2007, 44 percent of the 80 responding ARL member libraries reported they were delivering publishing services and another 21 percent were in the process of planning publishing service development. Only 36 percent of responding institutions were not active in this arena.

As of 2010, “Reshaping Scholarly Communication,” including electronic publishing, remains one of ARL's three strategic goals, suggesting that the initiative to develop such programs persists. The ACRL Research, Planning and Review Committee’s most recent review of trends and developments includes the prediction that “Libraries will continue to lead efforts to develop scholarly communication and intellectual property services.”

The increasing shifts from paper-based to electronic-only publication of scholarship more than justifies this trend. It is widely accepted that scholarly communication is reaching what Richard Johnson and Judy Luther identify as a “tipping point”:

The role of the printed journal in the institutional marketplace faces a steep decline in the coming five to ten years. Print journals will exist mainly to address specialized needs, users, or business opportunities. Financial imperatives will draw libraries first—and ultimately publishers also—toward a tipping point where it no longer makes sense to subscribe to or publish printed versions of most journals.
According to Lee Van Orsdel, in 2007 there were already over 2000 peer-reviewed open access journals, and approximately 600 interoperable open access repositories are in operation worldwide.6

Libraries can serve an important role in facilitating this transition and preserving important channels of scholarly communication. Journals that will no longer be economically viable as print publications will be required either to become electronic or cease. Even viable journals may prefer to develop their electronic versions using institutional, rather than private, resources. Further, e-publishing significantly lowers the cost of launching a journal, which opens opportunities to develop new ways of using journals to serve long-standing goals, such as using publication systems as educational tools. Libraries are well-positioned to support all of these goals by providing e-publishing services.

Much has been written about e-publishing in the context of libraries. This is evident from even a cursory survey of Charles W. Bailey’s “Scholarly Electronic Publishing Bibliography.”7 The literature regarding the development of open source software is even more extensive. However, the literature on the process of assessment is relatively sparse in both bodies of research; and there is virtually nothing on the problems that confront academic libraries in selecting open source software for e-publishing programs. So, for example, two of the most helpful accounts of building an electronic publishing program within an academic library context – at York8 and Cornell9, respectively – say little about how they chose the software they used. With this paper, we hope to contribute to filling this gap in the literature. We do not offer a one-size-fits-all approach or attempt to describe a comprehensive model for how to assess open source software for academic e-publishing programs. It is doubtful that there is any single “best” way of assessing the quality of such software. Instead the relevant criteria of assessment will depend crucially on the objectives of the particular department or institution. So, for example, if the objective is to begin a publishing program, then the criteria might be quite different from those that should be adopted if the desired objective is to create an open access knowledge base or journal repository. With this in mind, what we offer here is a description of the approach we used to evaluate specific software to be used within the context of our institution, as well as lessons we learned along the way. We hope these insights will prove useful to our peers in other institutions, even those involved in quite different endeavors, recognizing that their application will require adaptation to fit local circumstances.

Before proceeding to the description of our approach to software evaluation, there is an issue that should be addressed. In the time since we conducted our evaluation, OJS has emerged as the stand-out open source electronic publishing platform. Development has slowed or ceased on other projects, and new competitors have not emerged. Given this situation, the topic of evaluation might seem moot. We disagree. First, we think that our experiences — and the insights that we gained in evaluating e-publishing tools — will be helpful to colleagues engaged in the evaluation of a broad array of different sorts of software, especially open source software. Second, while OJS may seem like the only open source choice at the moment, this is unlikely to remain the case for long, given the rapid and unpredictable course of development in the open source community. Ultimately, libraries that undertake scholarly communication initiatives will need to make choices, and the insights we gained from our experiences can help with making those choices.
Evaluating Open Source Software

The evaluation of open source solutions poses different challenges from the evaluation of proprietary software. Leaving aside ideological factors that may lead libraries to prefer open source solutions, there are two primary considerations that militate in favor of such an approach. The first, and most obvious, consideration is economic. In contrast to commercial solutions, the initial starting costs of open source solutions are low because the software is free. Thus there can be good economic reasons to prefer open source software to commercial alternatives.

The second main consideration in favor of open source solutions concerns issues of institutional autonomy. Specifically, open source software enables a degree of autonomy for libraries not possible with commercial software since, more-or-less by definition; open source software can be supported and maintained by the end-user. Bugs can be fixed and additional features can be developed without assistance from the original developers. But this is a two-edged sword. Open source solutions generally do not come with professional technical support or training; and as a consequence libraries may find that the economic benefits associated with open source software—i.e. low, upfront costs—can be wiped out by the various expenses involved in fixing problems or learning cumbersome workflows.

In view of this, the task of making a good initial selection of open source software is an especially pressing one. Specifically, it is important to make a selection that is not merely sensitive to the virtues of autonomy and up-front savings; but also to the economic costs incurred during the use and maintenance of the e-publishing program. Or, to put the point another way, one needs to be sensitive to total cost of ownership (TCO); the direct and indirect costs throughout the entire life-cycle of the program. Open source software is “free” only in the initial instance: it can be installed and used without paying license fees. But the TCO can be very substantial.

**Total Cost of Ownership for Open Source Software: Some Considerations**

The concept of TCO has been around for many years and has been widely deployed in the IT industry, especially when evaluating the purchase of commercial computer systems. Moreover, evaluations based on TCO can be complicated, and an array of complex TCO evaluation models has been developed over the years.

We will not discuss the full models of TCO in detail, because the costs vary dramatically based on available resources and other factors. Each institution will need to assess the cost of different solutions based on the available skillsets, programs, and other factors.
relevant local factors. However, it is important that some overall assessment of cost—not just the initial purchase price—inform the evaluation process.

For example, in our evaluation, because we had already established a server and had trained staff, we did not include set-up costs. Instead we focused on the costs associated with publishing a first issue. Further, due to time constraints and considerations of transparency, we elected not to use a multi-variant model of TCO, but instead assigned a single cost variable to the assorted stages of the process. Thus, rather than constructing quantitative TCO models, the concept of TCO functioned more as a way of framing our overall deliberations: one that would be accessible to all those parties relevant to the selection decision.

In what follows, we begin with a brief review of the history of electronic publishing at the Ohio State University Libraries, and then describe the planning and implementation of our evaluation. Finally, we conclude with a brief account of how we presented our findings to the decision-making body.

**Case-Study**

**Background**

Prior to starting our evaluation, we had conducted some preliminary work, in 2007, to develop a dedicated e-publishing program that would complement existing open access and publishing efforts within the OSU’s DSpace-driven Institutional Repository called Knowledge Bank. The goal was to expand these efforts by using an e-publishing platform instead of the existing software used for the repository, DSpace. The publishing program began with one journal, which had been hosted on a commercial server using a custom-built, ASP-based interface. Initial efforts involved migrating this journal to the open source publishing software available to us called DPubS.

While using DPubS, we were able to establish our initial journal on our own server. However, in the process, we discovered some features of that platform that might limit the development of our e-publishing program. In addition, the development of other e-publishing solutions had advanced. The team therefore decided to revisit the question of which platform to build our program around and to conduct an extensive review and evaluation.

At the time of evaluation, the available staff time was limited to three part-time team members. Available skills included project management, software evaluation experience, IT infrastructure, and design of user interfaces and data gathering instruments. This team also consulted with two department heads who provided ongoing feedback and guidance.

In what follows, we describe the main stages of the evaluation process: choosing what systems to evaluate, specifying a relevant set of requirements, specifying hypothetical workflow models, gathering and analyzing information, and reporting findings to decision-makers.
Stage 1: Choosing which Systems to Evaluate

Though in 2008 several available open source e-publishing platforms existed, we decided to minimize the number of alternatives to evaluate. More specifically, based on a review of available assessments, expertise held by members of the team, and various other factors, we chose to restrict our evaluation to DPubS and Open Journal Systems (OJS).13

DPubS, an open source electronic publishing platform designed to enable models for scholarly communication and academic publishing, was originally created to support the cost-efficient distribution of serial literature in mathematics and statistics.14 DPubS uses a combination of Perl scripts and XSLT templates to manage content stored in a proprietary data system. Controls involve a combination of command-line and GUI interfaces.

Open Journal Systems (OJS), a journal management and publishing system, was developed by the Public Knowledge Project, through its federally funded efforts to expand and improve access to research.15 OJS uses PHP scripts and templates to manage content that is stored in a MySQL database. The program's functionality is controlled almost entirely through GUI interfaces.

Stage 2: Establishing Requirements

In order to assess an item of software it is important to be clear on what one’s needs are. Thus it is standard practice to start any software evaluation by describing in detail what the application needs to accomplish. In view of this, we began our evaluation by creating a set of system requirements, based on our expectations of what we wanted the tool to achieve.

The simplest approach to developing a set of system requirements would be to adopt some previously developed checklist for evaluating e-publishing software. Unfortunately, while checklists existed for evaluating digital library/institutional repository software,16 there were no similar guides for evaluating e-publishing tools, so we needed to develop a checklist of our own. One relatively time-consuming approach would have been to develop criteria from scratch. But we were able to identify standards of evaluation that had been applied to relevantly similar products that could be used as a basis for establishing our own set of requirements. Specifically, we took as a starting point the criteria for evaluating content management systems described in the Content Management Bible.17

Though content management systems share important similarities to e-publishing systems, there are also important respects in which they differ. For our purposes, then, there was a need to modify the checklist provided by the Content Management Bible. These modifications were largely based on our prior experiences of providing online publishing services, which included the publication of one active journal, and a longstanding collaboration with colleagues who were archiving back issues of OSU-related journals on an institutional repository known as the Knowledge Bank. The sample of the sorts of requirements that we adopted are contained in Figure 1.

Stage 3: Developing Work-Flow Models

One goal of our evaluation was to select a system that would allow us to provide services to stakeholders with quite different needs. Thus rather than assuming one “typical” client, we decided to assess how each system performed in addressing the needs
One goal of our evaluation was to select a system that would allow us to provide services to stakeholders with quite different needs.
diversity of the content the system needs to handle will be known up front and remain consistent. Thus, instead of versatility, these clients will prize consistency and reliability. Moreover, they will want robust finding tools, so that researchers can use the archives as a resource. Finally, they will want the journal to be as open to access as possible – with the exception of a handful of items restricted by copyright.

Considering such divergent models of digital publication not only helped clarify the specific features that our publishing system would need to possess, but also helped verify the importance of adopting a flexible system: one that could handle quite different kinds of work-flow.

Stage 4: Running the Models

Models A and B were framed at a high level of abstraction. In order to test the systems against these models, we produced a set of more fine-grained hypothetical workflow scripts associated with each model, and simulated the basic sub-tasks involved in publishing mock versions of a journal issue for each client model. The simulated activities included uploading content, content processing, content ingesting, and setting permissions. (For further details, see Appendix B.)

Given the specification of workflow scripts, we could test the relative ease or difficulty with which our two systems, DPUBS and OJS, handled the subtasks associated with Model A and Model B. More specifically, for each subtask, we created a worksheet pre-populated with the features most relevant to performance on that subtask. Each system was run on every subtask, and its performance assessed according to the extent to which it exhibited the desirable features. (See Appendix C for sample) The worksheets facilitated note taking on the experience of carrying out those tasks in each system, and most important, they allowed for a clear comparison of the two systems.

Stage 5: Analysis

We conducted two main forms of information analysis. First, we refined our qualitative observations to make it possible to make direct, point-by-point comparisons between the two systems. Second, we took the relative results of those point-by-point comparisons and, using a scoring system, produced a quantitative measure of each system’s comparative advantage.

Qualitative Comparison. Several considerations guided us through the process of systematizing our qualitative observations. First, we collated the raw information we had gathered into concise summaries that would make sense to our audience of decision-makers. Second, we refined our in-the-moment observations so that they would contain only observations about the functioning of the systems, as opposed to, for example, comments regarding how we felt about using the system at the time. Third, in the course of writing up our results, we identified gaps and oversights that needed to be fixed through additional observation. For example, we decided to provide an additional evaluation of each system’s ability to batch-import content, which was conducted by two technical members of the team, who each ran separate trials and gathered information that was then integrated into our overall descriptions of the two systems.

Quantitative Comparison. On the basis of our preliminary qualitative analyses, we produced an additional, overall, numerical cost-benefit analysis. Specifically, for each
feature in the initial qualitative analysis, we judged which, if either, system had a cost-benefit advantage — i.e. a greater cost-benefit value — over the other. This was then used to derive an overall cost-benefit advantage score for each system. A system with a high advantage score would tend toward low cost and high benefit. The chart below is a high-level representation of the feature-level comparisons at each stage of the process.

In addition to assessing the relative cost-benefit value for each stage in the publishing process, we also produced a set of overall advantage scores, which indicated how well each system performed in handling each of the two publishing models (Figure 3). We chose to express these scores as percentages because they represent the proportion of features for which a given system performs as well as or better than the other.

Stage 6: Reporting Results to Our Stakeholders

In our library system the responsibility for deciding which publishing platform to use for our publishing program fell to the Associate Director for Technical Services and Information Technology. As such, we needed to write our report not simply as a container of data, but in a way that would enable the administrator to make an informed decision about these systems without the benefit of direct experience with them. Many academic libraries employ an organizational model that separates the evaluation of the systems from the power to make the choices, so reporting plays an important but often underemphasized role in the decision-making process.
important but often underemphasized role in the decision-making process. Here, then, we would like to describe some of the strategies we used to make our report as clear and useful as possible. We do not offer the following as a universal recipe: each writing situation is unique, and what works for one audience may fail for another.

We used diverse strategies within our report, presenting aspects of our findings in a range of formats, including summary narrative, tabular summaries of subjective evaluation, tables of numerical data, and even a *Consumer Reports*-inspired table of comparative features. Decision-makers’ responses vary with different forms of information. Some prefer narrative descriptions, others prefer quantitative analysis. Some prefer paragraphs; others prefer bullet-points and diagrams. By presenting our results in a range of forms, we hoped to ensure that each of our readers would be able to access our findings in the way they found most comfortable. Here are some examples:

- A detailed full text requirement document, which included the general system requirements and the client model-specific;
- A table summarizing our findings about how each system performed on each aspect of each model’s requirements;
- A detailed description of the numerical cost-benefit advantage calculations, including a set of results tables and graphs;
- A brief topical overview of the main differences between the systems in the main areas of the evaluation;
- An executive summary of the most telling differences between the systems, described above, and with our recommendation that OSUL choose the Open Journal Systems.

Conclusion

In the above sections we discussed some of the central issues that arise in the evaluation of open source electronic publishing software. Moreover, we spelled out in some detail a recent case study involving the evaluation of such software within OSU Libraries.
The assessment methods we adopted allowed us to provide a detailed report to our stakeholders, and ultimately resulted in an informed and appropriate software selection. Specifically, OSUL opted for OJS, which has made the hosting of journals highly efficient, and has allowed staff to attend to the task of building a publishing program as opposed to focusing on technical matters. At the time of writing, we are hosting three journals in OJS using significantly less staff time than was previously required for a single journal.

We are also inclined to think that the process of evaluation had a highly desirable, though unintended side-effect. The time spent engaged in observation and analysis played double-duty as training; and by the time a final decision had been made regarding which system to adopt, we were already prepared to put that system into active use and to begin adding journals. Further, because we had studied the features and functionality of different systems, we developed a stronger understanding of the system’s design logic than might otherwise have been acquired.

The time spent engaged in observation and analysis played double-duty as training; and by the time a final decision had been made regarding which system to adopt, we were already prepared to put that system into active use and to begin adding journals.

Appendix A: General System Requirements

1. Introduction

Background: OSU Libraries is currently investigating new ways of collecting, sharing, disseminating and preserving scholarly knowledge. One central area of interest concerns the online publication of scholarly communications. With this in mind we are currently reviewing publishing systems such as: DPubS and OJS.

Objective of Review: The main objective of the review is to determine which, if any, of the systems suits the Libraries’ needs.

Assessment Criteria: Each system will assessed against a wide range of requirements intended to evaluate their sustainability and ability to accommodate different business models as they are developed.

Ruth Gallegos Samuels is Assistant Professor at the Ohio State University Libraries, e-mail at gallegos-samuels.1@osu.edu. Henry Griffy is Program Coordinator at the Ohio State University Libraries, e-mail at griffy.2@osu.edu.
Outcome of Review: The findings of the review will be used to inform stakeholders as to which system, if any, is appropriate for hosting an online journal publishing service at the Libraries.

2. Requirements

Submission of content
Functionality required by author or editor using the system

Requirement

Content authoring / editor specifics:
- Platform needs to be simple and easy to use
- Ability to paste/import from another document
- Ability to format text using: bold, italics, bullet points, font size, tables, footnotes, paragraphs, colour, indents, table of contents. Pre-set font.
- No limit on amount of text or size of document
- Ability to add images and other media
- Ability to create or transfer content from Word, Excel, PowerPoint, Dreamweaver
- Ability to add links documents

Metadata creation – Capture/create metadata (Author, subject, keywords, etc) for all content. Support for keyword indexes, subject taxonomies. Optional - Content classification for use in personalisation, search results, content targeting etc.

Metadata specifics -
- Metadata easily customised by publication e.g. what fields are required and what they may contain
- Ability to define optional and mandatory metadata – e.g. set data entry restrictions and validation
- Metadata easily managed by (Admin).
- Ability to edit/review document without need to re-add metadata
- Ability to batch load files with metadata

Linking – Ability to specify cross-links between items within the publication, and these must be stable against restructuring.
- Ability to link internally from one document to another.
- Ability to link to external web pages
- When reviewing/updating documents list of links to other documents should be available (links management)
Online management for Editorial process
Tools for managing and tracking the journal editorial cycle, including structure and review process

Requirement

**Access control** – role based authority to author / editor.
Administrators should be able to set up and manage roles and control allowable actions based on role. Be able to change roles of users easily.

**Workflow** ability to integrate with email for notifications (custom email messages for stages in workflow)
Ability to have a workflow set up for different stages: Submit, review, approve, and publish.
Validation process needed to publish content.
Custom workflows: Defined through XML (?)
Workflow to integrate with email for notifications (custom email messages for stages in workflow).
Be able to track content awaiting approval or review
Version control - Allow versions of content to be created. Added automatically each time content is published.
Reporting of version history – be able to recall older versions

**Security** - adequate security levels and audit trails must be in place to protect the integrity of the content.

**Administration functions** – to manage links, change metadata, archive / remove content.

Content mark-up and uploading
Tools for manipulating and managing the content (preparing for ingest)

Requirement

Batch loading
Be able to track content awaiting approval or review
Automation of coding process
Able to support any content file types simultaneously (e.g. picture in an html article)

**Special characters handler** – when uploading content able to identify special characters
Allow to re-use metadata added by author, editor

**Migration of existing content** - Be able to migrate content. (Mixture of html, Word, PDF files).
Content auditing: validation, links checker
Publishing and presentation / customisation

Requirement

Page templates – Be able to easily create templates to give a uniform look. Be able to modify templates when needed to alter appearance of entire publication or sections. Needs to be flexible.

1. Can design be edited using Dreamweaver?
2. Skills needed to create or modify templates – not too technical
3. Be able to make layout changes through a browser interface?
4. Wysiwig?

Support for multiple formats – be able to publish to multiple formats, such as HTML, printed, PDF

Personalisation – how flexible?

Usage statistics - comprehensive usage statistics

Usability – ease of use

Effective navigation - consistent, comprehensive and usable navigation aids.

Implementing Searching – what search functions?

- Full text searching
- Metadata (e.g. keyword) search

Integration with other systems – e.g. library catalogue, DSpace - KB

Accessibility - Needs to be accessible from all size screens – small and large at work, home and laptops

Subscriber Access Control

Requirement

Batch uploading – information

IP and password controls

Self managing of passwords – change / reset and email notification

Flexible to accommodate various business models

Customisation – access by content (e.g. make one issue open access or one article depending on user needs.

Ability to sell and purchase individual articles

Ability to manage rights and royalties – keep track on copyright and price royalties (off-prints/ ongoing access for contributors)
Appendix B: Model-based Requirements Used in Evaluation

Overview: These models are designed to approximate realistic publishing models that might well be encountered. OSU’s existing publication, DSQ, generally falls somewhere between the two evaluation models. It should thus serve as a useful point of reference for the evaluation process.

Model A: The general idea behind Model A is a group that is publishing new content, with a high degree of diversity (some already known, other choices remaining to be made). They will want to use the system as much as possible to guide and regulate the preparation and processing of content. They will want maximum control of the process, along with a high degree of flexibility in the format and presentation of content. They will also want robust and flexible access controls, as they intend to run the journal as a revenue-generating endeavor.

Model B: Model B represents a retroconversion project for a long-term publication that has ceased publication. They will not need the system to produce content – or even to process content -- and the diversity of the content the system needs to handle will be known up-front and remain consistent. Instead of diversity, these clients will want consistency and prize reliability. They will want robust finding tools, so that researchers can use the archives as a resource. Along these lines, they will want the journal to be as open to access as possible -- with the exception of a handful of items restricted by copyright.

In describing these models, the goal was not to produce the most extreme contrasts possible, but rather to describe two realistic but distinct publication models that would invoke as many important features of the systems as possible.
Standard Workflow Model
With each of the models, the workflows generally break down into about a half-dozen stages, which are defined as follows:

**Content Creation:** In traditional publishing, this would constitute all workflows preceding camera-ready pages. This mainly involves collaborative work among partners of the library, such as composition, editing, reviewing, copyright clearance, etc. Generally speaking, the prototypical format for the results of this workflow would be word-processing documents and/or image files.

**Content Processing:** This refers to system-specific steps required to convert the "camera-ready" content into web-deliverable form. The work primarily involves using HTML and/or another scheme to mark up the material. Planning for this work involves designing the mark-up scheme to be used for a given publication, in order to lay the groundwork for later styling using CSS.

**Content Ingest:** This refers to the process of importing web-deliverable content into the system and making it available for delivery to end-users by the system.

**Content Organization:** This refers to the grouping of individual items into content units. In practice, this will probably most often mean building Tables of Contents for Issues. However, some clients may want to provide other sorts of paths to their content. Also, different systems will accomplish this task in different ways.

**Site Organization:** This refers to the organization of "static" content: standard pages (i.e. Contacts, Submission Guidelines) and custom/constant pages (History of the Publication) This also refers to the procedures available to end-users for managing site functionality (logging in) and/or customization (increasing font).

**Access Control:** This refers to the fundamental task of enabling and/or preventing the viewing of any individual bit or group of content by any given end-user.

**Interactivity/Feedback:** This is a bit more speculative, at this point, but clients have expressed some desire to provide ways for users to respond to the content of a publication. At the low-end, DSQ currently provides e-mail addresses for many contributors. More ambitiously, many online publications provide comment sections, forums, and even areas for uploading additional content.
Standard Workflow Model
With each of the models, the workflows generally break down into about a half-dozen stages, which are defined as follows:

<table>
<thead>
<tr>
<th>Model</th>
<th>A: Client-driven New Content</th>
<th>B: Retrocon Plus</th>
</tr>
</thead>
<tbody>
<tr>
<td>Content Creation:</td>
<td>• clients will take responsibility for all aspects of content creation</td>
<td>• content will consist of back issues in both paper and electronic form, but just a few known and predictable formats.</td>
</tr>
<tr>
<td></td>
<td>• clients want to use the system to collaborate, communicate, and track all preparatory work:</td>
<td>• clients will provide both the content and its accompanying metadata</td>
</tr>
<tr>
<td></td>
<td>• submissions</td>
<td>• but metadata will be flawed and need some post-facto correcting</td>
</tr>
<tr>
<td></td>
<td>• reviewing</td>
<td>• minimal linking, mostly internal</td>
</tr>
<tr>
<td></td>
<td>• revisions (w/ versioning)</td>
<td>• clients may want to add an introductory/descriptive piece for each issue ingested</td>
</tr>
<tr>
<td></td>
<td>• approvals</td>
<td>• copyright tracking: they need copyright tracking mechanism to track copyright clearance process</td>
</tr>
<tr>
<td></td>
<td>• clients would like the option to use a built-in editor -- or not</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• clients want ability to use multiple formats (text, image, etc.)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• individual items may be different (html, pdf, etc.)</td>
<td>• little to no content mod required</td>
</tr>
<tr>
<td></td>
<td>• individual items may combine multiple formats (html w/ images, e.g.)</td>
<td>• metadata review/correction: ideally, clients would like built-in metadata validation</td>
</tr>
<tr>
<td></td>
<td>• content will contain multiple links to content within the publication, as well as other websites, etc.</td>
<td>• standardized processing, applied to large amount of content</td>
</tr>
<tr>
<td></td>
<td>• metadata: clients will produce metadata (mostly in the process of preparing items for publication) so:</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• need ability to add, edit, and delete field contents</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• want ability to create and define fields</td>
<td></td>
</tr>
<tr>
<td>Content Processing:</td>
<td>• maximum recycling of metadata from the content creation stage</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• ability to handle multiple formats in variable configurations</td>
<td></td>
</tr>
<tr>
<td>Model</td>
<td>A: Client-driven New Content</td>
<td>B: Retrocon Plus</td>
</tr>
<tr>
<td>-------</td>
<td>---------------------------------------------------------------------------------------------</td>
<td>---------------------------------------------------------</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Content Ingest:</td>
<td>• issue-specific variation</td>
<td>• batch loading</td>
</tr>
<tr>
<td></td>
<td>• batch-loading desired</td>
<td>• consistent process</td>
</tr>
<tr>
<td>Content Org’n:</td>
<td>• clients want maximum control and flexibility over the way content is arranged and provided</td>
<td>• simple organization scheme, adapted from existing content</td>
</tr>
<tr>
<td></td>
<td>• clients want to customize tables of contents for individual issues</td>
<td>• main feature: consistency and ease of use</td>
</tr>
<tr>
<td></td>
<td>▶ sequence of items</td>
<td>• issue-level browsing required</td>
</tr>
<tr>
<td></td>
<td>▶ division into sections</td>
<td>• alternative finding tools (indexes, lists of authors, titles, etc.) more important</td>
</tr>
<tr>
<td></td>
<td>▶ choosing what metadata is visible to end-users</td>
<td>• searching tools: must be robust: by issue/date, keywords, subjects, item type, other metadata</td>
</tr>
<tr>
<td></td>
<td>• clients want variable definition of what constitutes an issue -- items may appear in more than one(?)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• each issue may follow a different organizational scheme</td>
<td></td>
</tr>
<tr>
<td>Site Org’n:</td>
<td>• clients want to create and customize their static/info pages</td>
<td>• clients want to provide raw content to be converted into minimal static pages</td>
</tr>
<tr>
<td></td>
<td>• clients want to manage the navigation elements (to add and remove items from menus, etc.)</td>
<td>• clients will submit change-requests (new address, e.g.) for us to implement, but want quick turnaround</td>
</tr>
<tr>
<td></td>
<td>• clients want site versioning -- recording what the site looked like at specific moments in time</td>
<td></td>
</tr>
<tr>
<td>Presentation/Style:</td>
<td>• clients want control over style accessibility: must be easy to maintain user-based and device-based standards</td>
<td>• basic goal is to replicate their existing style</td>
</tr>
<tr>
<td></td>
<td>• customizable on at least three levels:</td>
<td>• clients want a clear, consistent, standard interface</td>
</tr>
<tr>
<td></td>
<td>▶ publication-level (e.g., all items use same font)</td>
<td>• minimal customization</td>
</tr>
<tr>
<td></td>
<td>▶ issue-level (e.g., “the blue issue”)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>▶ item-level (e.g., “special effect for a given poem”)</td>
<td></td>
</tr>
<tr>
<td>Model</td>
<td>A: Client-driven New Content</td>
<td>B: Retrocon Plus</td>
</tr>
<tr>
<td>-------</td>
<td>-----------------------------</td>
<td>------------------</td>
</tr>
<tr>
<td>Access Control:</td>
<td>• subscription default</td>
<td>• open-access default</td>
</tr>
<tr>
<td></td>
<td>• limited open-access</td>
<td>• limited by gaps in copyright</td>
</tr>
<tr>
<td></td>
<td>o select issues</td>
<td>clearance and permissions</td>
</tr>
<tr>
<td></td>
<td>o 2 items per each issue</td>
<td>(say, 12 items not cleared – or</td>
</tr>
<tr>
<td></td>
<td></td>
<td>permission restricted to local</td>
</tr>
<tr>
<td></td>
<td></td>
<td>OSU use)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• detailed statistics: how many</td>
</tr>
<tr>
<td></td>
<td></td>
<td>times each article is accessed,</td>
</tr>
<tr>
<td></td>
<td></td>
<td>whom</td>
</tr>
<tr>
<td>Interactivity/Feedback:</td>
<td>• client wants maximum possibility</td>
<td>• none requested</td>
</tr>
<tr>
<td></td>
<td>for feedback:</td>
<td>• possible: form for end-users to</td>
</tr>
<tr>
<td></td>
<td>o end-user should be able</td>
<td>describe content they looked for</td>
</tr>
<tr>
<td></td>
<td>to contact each contributor</td>
<td>but were not able to find</td>
</tr>
<tr>
<td></td>
<td>o possible comments section</td>
<td></td>
</tr>
<tr>
<td></td>
<td>o ideally, threaded</td>
<td></td>
</tr>
<tr>
<td></td>
<td>comments</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• they have floated the idea of an</td>
<td></td>
</tr>
<tr>
<td></td>
<td>issue consisting primarily of end-</td>
<td></td>
</tr>
<tr>
<td></td>
<td>user contributed content</td>
<td></td>
</tr>
</tbody>
</table>

**Appendix C: Worksheets Used to Compile Evaluation Data**

**Evaluation Tool for Digital Publishing System**

<table>
<thead>
<tr>
<th>System:</th>
<th>DPubS/OJS</th>
<th>Model:</th>
<th>Date:</th>
</tr>
</thead>
</table>

**Stage: [Preliminary Work / Establishing the Publication]**

**Req:**

**Workflow:**

- **Evaluation:**
- **Time:**
- **Comments:**

**Stage: [Content Creation]**

**Req:**

**Workflow:**

1. **Evaluation:**
1. **Time:**
1. **Comments:**

1. **Evaluation:**
1. **Time:**
1. **Comments:**
Stage: [Content Processing]

Req: 
Workflow: Evaluation: Time: _____________
1. Comments: 

Req: 
Workflow: Evaluation: Time: _____________
1. Comments: 

Stage: [Content Ingest]

Req: 
Workflow: Evaluation: Time: _____________
1. Comments: 

Req: 
Workflow: Evaluation: Time: _____________
1. Comments: 

Stage: [Content Organization]

Req: 
Workflow: Evaluation: Time: _____________
1. Comments: 

Req: 
Workflow: Evaluation: Time: _____________
1. Comments: 

Stage: [Site Organization]

Req: 
Workflow: Evaluation: Time: _____________
1. Comments: 

Req: 
Workflow: Evaluation: Time: _____________
1. Comments: 

Stage: [Access Control]

Req: 
Workflow: Evaluation: Time: _____________
1. Comments: 

This mss. is peer reviewed, copy edited, and accepted for publication, portal 12.1.
Req:
Workflow: Evaluation: Time: ________________
1. Comments:

Stage: [Interactivity/Feedback]

Req:
Workflow: Evaluation: Time: ________________
1. Comments:

Req:
Workflow: Evaluation: Time: ________________
1. Comments:

Notes


