TRADING PAGES: USING SOFTWARE TO MANAGE EDITORIAL WORKFLOWS AT SMALL CULTURAL MAGAZINES

by

Wesley Fok
Bachelor of Arts, Queen's University, 2004

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

MASTER OF PUBLISHING

In the
Faculty of
Arts & Social Sciences

SIMON FRASER UNIVERSITY

Summer 2008

Approval

Name: **Wesley Fok**

Degree: **Master of Publishing**

Title of Project: **Trading Pages: Using Software to Manage Editorial Workflows at Small Cultural Magazines**

Supervisory Committee:

_____

John Maxwell, PhD
Senior Supervisor
Assistant Professor, Master of Publishing Program
Simon Fraser University

_____

Rowland Lorimer, PhD
Supervisor
Professor and Director, Master of Publishing Program
Simon Fraser University

_____

Megan Griffith-Greene
Industry Supervisor
Editor-in-chief
Shameless Magazine, Toronto, Ontario

Date Approved: _____

Abstract

This report is a study of editorial workflow practices and the use of editorial workflow software at small Canadian magazines. Seven magazines were surveyed about their current staff and working environments, their editorial workflow practices, and the use of software to manage workflow. The magazines were also asked to evaluate their current workflow and explain their attitudes towards workflow software. This report also reviews several existing workflow software packages with an eye towards their suitability for a small magazine environment, including a discussion of cost, feature set, and ease of use. Based on the reviews and the requirements derived from the workflow models of the magazines surveyed, this report finally explores the possibility of developing new workflow software tailored for small magazines, describing basic requirements and guidelines for future software development.

Keywords: editorial workflow, magazine publishing, small magazines, workflow software

Acknowledgements

A small group of distinguished individuals helped a great deal with this report, and are worthy of praise and adulation.

There are the members of my program who provided vital guidance and support: my academic supervisors, John Maxwell and Rowland Lorimer; Jo-Anne Ray; and the MPub class of 2004.

There are the magazines who enthusiastically donated their experiences, insights, and opinions: Megan Griffith-Greene, my industry supervisor; and the editors of *Briarpatch*, *Queen's Alumni Review*, *Spacing*, *Taddle Creek*, *This Magazine* and *Up Here*.

Finally, there are the many friends and well-wishers who served as a collective sounding board and offered their support in numerous ways.

Table of contents

List of figures

Chapter 1: Introduction

This report is a study of current editorial practices at seven small Canadian magazines, undertaken with an eye towards exploring the use of editorial workflow software in small magazine environments. The intent of the report is to provide information relevant to the development of a low-cost software package aimed at facilitating the editorial workflow at magazines with limited resources and small staff.

Traditionally, small magazines have suffered from a number of situational issues that, put together, can make the continued survival and prosperity of an independent publication rather troublesome. Small magazines rely on relatively small staffs consisting largely or entirely of part-time or volunteer workers. Small magazines also lack many of the resources larger publishers may have—some of the magazines that took part in this study did not even have an office from which to base their operations, let alone other niceties like company-owned computer equipment.

Several of the magazine editors that contributed to this study all expressed the same basic problem with their job: too much work to do, not enough time to do it in. Between editing a magazine, keeping the rest of the staff on task, and managing various non-editorial duties on the side, there is a lot for an editor-in-chief to handle. Other editors, who often work part-time or volunteer, must perform their various editorial duties while maintaining employment elsewhere. Confusion with scheduling and coordination can be an issue for small magazines, as can the sheer amount of work an editorial staff has to complete in order to get an issue out the door.

My personal experience with publication workflows first came while running a university newspaper. As a student-run organization with a somewhat limited budget, the newspaper operated in a general atmosphere of chaos, especially on production days. Our editorial workflow was similar to that of a small magazine. Editors assigned stories to writers, who then sent in first drafts via e-mail. Those drafts went through several editing stages before going to layout and copyediting. Once the layouts had been proofed, they went to press. What made the process more difficult was keeping track of everything: where articles were, what stage of editing each article was at, and who was supposed to be editing what at any given moment. All these things we kept track of on paper, on whiteboards, and in our heads—a system that, while workable, was prone to error. We would often misplace hard copies of edits, occasionally forget to pass an article to an editor, or lose a document and have to re-enter edits.

The city newspaper I interned at could not rely on such a fragile system. Instead it used a software system called Quark Publishing System (QPS) to keep track of articles and layouts. Because QPS tracked the various editing stages of an article, every editor who was supposed to see an article was guaranteed to see it; because QPS handled document storage on its own, overwriting or misplacing the most current version of an article was impossible. QPS streamlined and simplified the editorial workflow of the newspaper by taking the burden of keeping track of everything off the various editors. As a labour-saving and stress-preventing device, QPS was a revelation. Though we did not have the budget at our university paper to adopt QPS ourselves—the $25,000 price tag of a similar system, Softcare's K4, would have eaten up a substantial portion of our budget at a time when we were losing money[1]—it was clear that such editorial workflow software packages represented a major improvement on our editorial processes.

[1] See the discussion of systems like QPS and K4 on page 45 for details on pricing.

The working environments at small magazines more closely resemble the barely contained chaos of a university newspaper than the managed calm of a city newspaper. The magazine case studies in the next chapter will show that a number of small magazines also use manual methods for keeping track of the editorial process—the same methods that led to missing files, missed edits, and general disorder. These magazines rely almost entirely on the innate organizing abilities of their staffs to stay on schedule. The added rigour of editorial workflow software could potentially relieve magazine staffs of this burden, leaving them to focus on the tasks of editing, designing, and producing a magazine. Many of the advantages of large, monolithic systems like QPS can be recreated today on a much smaller scale, and for a much lower cost to users. Taking advantage of the lower requirements of small magazines, as well as the low cost of entry for web-focused database solutions, it should be possible to build a low-cost workflow system designed to handle the needs of small magazines. The technology is already here—it is just a matter of designing an application to take advantage of it.

This report is split into three sections, covering the current workflow practices of small magazines; the current state of workflow software; and the possibilities for future workflow software development. First, this report will examine how various small Canadian magazines go from first draft to finished product, and establish their use of and attitudes toward software to manage editorial workflow. This will be followed by an examination of the current crop of products available to small magazines wishing to improve their workflow. This report will also explore some of the issues facing small magazines and their adoption—or lack thereof—of workflow software. Why have some small magazines not adopted editorial workflow software, and how can software packages be improved to better fit the needs of small magazines?

Finally, this report will briefly look towards the possible creation of a new workflow software package, and how we might get there from our current position. What lessons can be learned from the workflow practices and working environments of small magazines? How well do current software packages cover the market, and what kind of opportunity exists to modify an existing package or create a new application specifically targeted at small magazines?

Chapter 2: Magazine case studies

The case studies in this chapter were compiled from questionnaires sent to and interviews conducted with the editors of the participating publications between September and December 2007, after an initial request for participants sent to approximately 35 small magazines across Canada. Of the magazines queried, the *Queen's Alumni Review*, *Taddle Creek*, *Briarpatch* and *Shameless* all responded via e-mail questionnaire, while *Spacing*, *This Magazine* and *Up Here* all responded via phone. All magazines were asked the same basic set of questions covering basic information about the magazine; the makeup of magazine staff and pool of contributors; long-term and issue-by-issue planning; the editorial workflow, and pros and cons of such; and the use and attitudes towards workflow software.

The seven magazines that responded covered a variety of subjects, embodied a wide range of workplace environments, and claimed various levels of success with the different methods they had come to use in managing the production of an issue. What was especially interesting, however, was that despite all these differences, all seven magazines shared a common concept of what an editorial workflow should look like, both in terms of the people involved and the stages of editing and production. All utilized a fairly linear workflow model, and most relied on a single person—often the editor-in-chief[2]—to keep track of the whole process.

[2] A brief note about terminology: while the various magazines use different titles for the role of senior-most editor (usually Editor), for clarity's sake I have used the term "editor-in-chief" to differentiate from "editor," used often to refer to anyone editing a document, and "handling editor," the editor responsible for the substantive editing of an article.

Queen's Alumni Review

- **Location:** Kingston, Ontario
- **Schedule:** quarterly
- **Circulation:** 106,000
- **Audience:** mostly Queen's University alumni
- **Content:** news and features about Queen's University and prominent alumni; monthly list of alumni marriages, deaths, achievements, etc.
- **Respondent:** Ken Cuthbertson, editor-in-chief

The *Queen's Alumni Review* is one of the more well-supported magazines in this study. It has an editorial staff of six people, consisting largely of part-time paid editors and designers, working out of an office on the Queen's University campus. Its editorial workflow model can be characterized as linear, with two editors handling the bulk of the process and articles generally handled by a single person as it passes through each stage. The magazine uses no software to assist its workflow. The manual system the magazine used was judged sufficient for their needs.[3]

Most of the magazine's six editorial staff works on a part-time basis. The editor-in-chief is the only full-time editorial staffer. Also included in the core staff are a half-time editor responsible for maintaining the Keeping in Touch class notes section of the magazine, an art director associated with Queen's Marketing and Communication, an editor emerita who works on a freelance basis, and a freelance proofreader/copyeditor. The last two staff members work remotely. The half-time advertising coordinator also manages production, and so can be considered a member of the editorial staff. A given issue of the *Alumni Review* tends to use a dozen or so contributors for articles,

[3] Ken Cuthbertson (editor-in-chief, *Queen's Alumni Review*), e-mailed questionnaire, November 4, 2007.

photographs and illustrations. A significant portion of an issue's photography comes from stock sources such as the Queen's archives and file photos.[4]

Generally, each issue of the *Alumni Review* includes a cover story of four to five pages, two to three shorter features, and several single-page features. Front-of-book and back-of-book material consists of a letters section of two to three pages, a Eureka! section for campus research news, a Gazette page for campus news, and an At the Branches section for alumni branch news. Finally, the Keeping in Touch section takes up roughly a quarter to a third of the magazine's page count, and consists largely of personal news from Queen's graduates in a classified-style layout.[5]

Planning for a given issue generally takes place during meetings held by the editor-in-chief and includes the ad coordinator, the notes editor, the director of communication at Queen's and a representative from the alumni association. The assembled editorial staff and university representatives review cover and feature stories during these meetings. The magazine commissions a significant number of stories but also accepts submissions, which are examined and accepted by the editor. There is also an art meeting where the editor, art director and communications director discuss art treatments for the cover and interior pages.[6]

The editing process at the *Alumni Review* is fairly protracted, as generally only the editor-in-chief and the editor emerita handle the substantive editing of a piece. The editor-in-chief reviews submissions and discusses them at a story meeting. Once an article has been selected for publication in the issue, the editor-in-chief makes a substantive edit and adds a headline, deck, relevant photos and photo credits, after which the editor-in-chief passes the piece on to the editor emerita via e-mail for further

[4] Cuthbertson, e-mailed questionnaire; Ken Cuthbertson, supplementary e-mail correspondence, July 30, 2008.

[5] Ibid.

[6] Ibid.

comments, suggestions and proofreading. Once the piece meets the editor emerita's approval, it is passed on to the art director and placed into layout. The editor-in-chief and notes editor proof all layouts, followed by the proofreader. Once the editor-in-chief inputs the final changes, the article is ready for the printers.[7]

The editor-in-chief rated the current workflow system as very good "after many years of trial and error." He generally saw it as reliable and straightforward, with very few problems stemming from the process itself, though he mentioned that writers missing deadlines or staff out sick could cause delays. The *Alumni Review* does not use software to help manage the magazine's workflow. He felt adopting workflow software was unnecessary and, indeed, could be detrimental to the magazine's operation.[8]

Taddle Creek

- **Location:** Toronto, Ontario
- **Schedule:** twice yearly
- **Circulation:** 1,500
- **Audience:** Toronto-based readers with an interest in local literature
- **Content:** prose and poetry, primarily from Toronto authors
- **Respondent:** Conan Tobias, editor-in-chief/publisher/art director

Taddle Creek is one of the smallest of the seven magazines surveyed. Of the seven core editorial staffers, none are full-time; everyone is either volunteer or freelance. The magazine also has no central office. *Taddle Creek*'s editorial process is linear, and much of the editing and layout work is performed by two people. The magazine does not use any software to manage their workflow; instead everything is managed on paper. Despite

[7] Ibid.

[8] Ibid.

this, the editor-in-chief felt no particular need or desire to change the system after a decade of use.[9]

Conan Tobias is the editor, publisher and art director of the magazine; there is also an associate editor, a copyeditor, a proofreader, and two contributing editors "who do this and that." Finally, to assist with layout and design issues, there is a contributing artist as well. Each issue tends to include about 20-25 contributors, mostly writers and poets alongside a handful of cartoonists and illustrators, and generally a single photographer. *Taddle Creek* has no office, which means production of an issue tends to occur in people's homes and via e-mail.[10]

The fiction content makes up the bulk of the magazine, and consists of six to eight pieces of prose that vary in length from a few hundred words to 4,000 words, six to eight pieces of poetry, and a comic strip. The non-fiction content of the magazine is made up largely of shorter pieces: an author profile; a non-fiction essay; a comic strip; a short feature about the work of a visual artist; and several recurring features like reviews and an editor's note. Aside from the comic strip, each piece of fiction is accompanied by illustrations, and the author profile includes a photograph.[11]

As *Taddle Creek* does not have theme issues, the editorial mix tends to remain the same from issue to issue, simplifying the planning process. The material is mostly solicited, though the magazine maintains a slush pile that both the editor-in-chief and contributing editor read on a regular basis. Non-fiction material is generally assigned. The planning process in general appears to be ad hoc, where for a given issue "we solicit a few pieces, see what the slush pile has provided us with, then solicit whatever else we need." Because the magazine places relatively little importance on unsolicited

---

[9] Conan Tobias (editor-in-chief, *Taddle Creek*), e-mailed questionnaire, November 24, 2007.
[10] Ibid.
[11] Ibid.

submissions, and perhaps because of the small staff, there is little structure to the magazine's review process for submissions. Submissions are tracked on paper and decisions are made between the editor-in-chief and the contributing editor.[12]

As with the *Queen's Alumni Review*, the editorial process at *Taddle Creek* involves relatively few steps and focuses on the editor-in-chief as the hub of all editorial interactions. Pieces arrive at *Taddle Creek* in electronic form, usually Word documents but occasionally in text files or in the body text of an e-mail. Both the editor-in-chief and associate editor read through and write up notes for the author regarding changes to the piece. Once substantive editing is finished, the editor-in-chief fact-checks and copy-edits the piece (occasionally other people perform the fact checking). Then the piece is passed on to the freelance copyeditor, and changes are again discussed with the author.[13]

After the copy-editing stage, the piece is placed into layout and fitted. After this point, changes tend to be minor and so authors are sent PDF copies along with notes explaining the changes. Once the author has signed off on the PDF layout, the piece is proofread by the editor-in-chief and the freelance proofreader. Final changes are incorporated and the piece is ready for press.[14]

The process is relatively linear, with only one person handling a piece at any given time. In this way, the editor is able to keep track of everything on paper. *Taddle Creek* has used the same editing and production process for a decade and Tobias saw no need for change, as their system has worked perfectly for the magazine. As such, he neither has looked into using software to manage the magazine's workflow nor sounds particularly interested in adopting any. He also said that without a central office or

[12] Ibid.

[13] Ibid.

[14] Ibid.

computer infrastructure, it would be difficult to set up editorial workflow software in the first place.[15]

Up Here

- **Location:** Yellowknife, Northwest Territories
- **Schedule:** eight times per year
- **Circulation:** 27,000
- **Audience:** readers interested in, living in, or once a resident of the Canadian North
- **Content:** news and features about Northern life; profiles of Northerners; travelogues and photo essays
- **Respondent:** Aaron Spitzer, editor-in-chief

Up Here is a small operation, but has more resources at its disposal than most of the other magazines surveyed. Six people make up the editorial staff, all full-time and salaried. The magazine shares an office with other magazines owned by the same publisher. The workflow system *Up Here* uses is linear and manual, though the magazine uses rudimentary forms of computer tracking like changing filenames to represent an article's status. The magazine has not looked at utilizing software for workflow assistance, but the editor-in-chief believed there were potential benefits. The magazine's current workflow performs acceptably but is somewhat fragile due to staffing and planning issues.[16]

Editorial operations are handled by a relatively small staff, though the staffs of the other magazines published in the Up Here Publishing offices (at the time of the survey, *Canadian Diamonds* and *Far North Oil and Gas*) assist with various aspects of *Up Here*'s production in an informal manner. There is an editor-in-chief overseeing all editorial operations, as well as a senior editor and an associate editor handling the text of

[15] Ibid.

[16] Aaron Spitzer (editor-in-chief, *Up Here*), phone interview, November 2, 2007.

the magazine. On the art and production side, there is a photo editor/researcher and an art director, as well as a production assistant. The magazine has no official copyeditors or proofreaders, those duties being handled by the rest of the staff and by the staff of the other magazines as needed. The senior editor handles more of the traditional editing duties, while the associate editor takes on a wider variety of tasks such as writing small pieces for the magazine, doing photo research and maintaining the magazine's blog.[17]

The magazine uses about 20 contributors per issue, the bulk of whom are photographers. Most photographs in an issue are contributed. On the writing side, freelance writers handle the magazine's feature articles and columns, while magazine editors handle all front-of-book material. While the magazine does accept unsolicited submissions, the vast majority of the magazine's content is commissioned, with some stories developed from pitches.[18]

Planning for the magazine's publishing schedule occurs in several stages. For the 2008 publishing year, initial planning began in August 2007, with a draft framework built by the whole staff. It is during this initial planning that issue themes are decided on, though generally they do not change very much from year to year—for example, the January and summer issues tend to be devoted to travel. After the framework has been laid out, the editor-in-chief also does a fair amount of planning for each individual issue at the beginning of each issue's two-month publishing cycle. Aside from consultations with the senior editor, most of this planning is done by the editor-in-chief, and tends to relate mostly to day-to-day matters. One major exception is art direction. The editor-in-chief, art director and photo editor sometimes have a graphic meeting in preparation for each issue where they discuss possible art treatments.[19]

---

[17] Ibid.

[18] Ibid.

[19] Ibid.

During the editorial and production cycle, each of the three editors take on a portion of the stories in an issue, and serve as the author's contact and story editor during the initial stages of editing. Most of the issue is split between the editor and senior editor, with a smaller portion falling under the responsibility of the associate editor. Once the first draft of an article arrives at the magazine via e-mail, the handling editor gives it an initial read and edit, and corresponds with the author further. After several weeks of refinement, the article then goes to the editor-in-chief for a second read; for articles where the editor-in-chief is also the handling editor, the second reader is the senior editor. At this point substantive edits are relatively light.[20]

Once the second stage of editing is complete, the article is placed in the art director's folder on the magazine's file server. After the art director places articles in layout, the pages go through a copyediting and proofreading stage. At least two different people look at pages during this stage, usually the senior editor and the editor-in-chief of one of Up Here Publishing's other magazines; the copyediting process tends to be informal. The art director inputs whatever copyediting changes need to be made, and the editor-in-chief and associate editor proofread the whole issue one last time before press.[21]

During the production of an issue, editors send electronic copies of an article to each other either by e-mail or by using the magazine's file server; there are no specific guidelines for file handling and so which method gets used is essentially random. Once an article enters layout, changes to the article can only be made on the single office computer with InDesign installed, which simplifies the process of tracking an article's process but also makes it difficult to edit the article, since that computer must do double duty as both a layout computer and an input computer for copyediting and proofreading

[20] Ibid.

[21] Ibid.

changes. Substantive edits take place in Word, while copyedits and proofreading changes are marked on paper and merged into the InDesign layout.[22]

Articles are tracked in the system in two ways. The first method is by giving each article on the file server a filename prefix indicating who should look at the file next. For example, if the associate editor needs to look at an article, its filename will be marked with her initials. Because files are not always transferred to other staffers via the file server, and because mistakes are occasionally made in labelling, the editor-in-chief also keeps track of all articles himself. He is able to do this by virtue of having a small staff and a limited number of articles per issue. As for the other people involved in the process, they have a limited perspective on an issue's progress; each editor and the art director has a good idea of what they need to do, but generally they do not keep track of every article's status.[23]

Up Here's editorial workflow is sufficient, the editor-in-chief felt, but could definitely be improved further. Most of his criticisms have to do with what he felt was a lack of structure and pre-planning, as well as staffing concerns leading to a lack of personnel redundancy and high workloads for staff members. He also believed the tight production schedules and the lack of a buffer of stories made the process fragile; the editor-in-chief wished he was an extra issue ahead, so that in November the magazine staff would be preparing the January/February issue instead of the December issue, and assigning stories for the March issue as well. "We're surviving," he said, "but at any given time we're hanging by a thread."[24]

Despite these concerns, *Up Here* has never investigated the possibility of introducing workflow management software to the magazine's operations. There are

[22] Ibid.
[23] Ibid.
[24] Ibid.

several factors for the lack of investigation, including a "passive satisfaction" with the current system (in that it works well enough, if not optimally), a lack of research time, and a lack of understanding regarding the possible benefits of such software. Two areas where software was seen to offer possible improvement include some form of version control to keep older drafts available for review, and some method of synchronizing web and print content. Web versions of print articles take their text from the Word documents, and thus copyediting and proofreading changes made in InDesign are never merged into the web articles.[25]

This Magazine

- **Location:** Toronto, Ontario
- **Schedule:** six issues a year
- **Circulation:** 5,000
- **Audience:** politically-aware progressive/liberal readers, split between middle-aged, long-time readers and younger, twenty-/thirty-something readers
- **Content:** politics, arts and culture from a politically progressive perspective
- **Respondent:** Jessica Johnston, editor-in-chief

This Magazine is a well-established magazine that stakes out the middle ground of the magazines surveyed. The editorial staff consists of about fifteen people, with the editor serving full-time and the others on a volunteer basis. The magazine has a central office, though it is used largely by a small subset of staff. *This* uses a linear workflow system, managed largely by the editor-in-chief; workloads tend to be more dispersed than the magazines featured previously, who rely on smaller staffs and lack section editors. The magazine does not currently utilize software to help manage their workflow,

[25] Ibid.

nor future plans to adopt any. The current workflow is stable and effective, but the magazine is open to the possibility of using software.[26]

The magazine's editorial staff straddles the gap between volunteer and full-time staff as well as telecommuter versus office worker. The editor-in-chief is the magazine's main full-time editorial staffer, and works out of the *This Magazine* office in downtown Toronto along with the art director and two editorial interns, who perform a variety of duties but serve mainly as the magazine's fact-checkers. There are also four section editors, a photo researcher, and five to six people serving as copyeditors and proofreaders per issue; all of these individuals are volunteers and mostly work outside the office. About 25 writers contribute articles to each issue. The number of illustrators and photographers varies more, with an average of about three to four illustrators and two photographers per issue. All material is solicited, with the vast majority coming from assignments and relatively few coming from writer pitches.[27]

The editorial mix consists of four sections, split in half by the feature well. The front-of-book material consists of the This & That section, which averages about five pages and contains articles of between 100-600 words, and three columns of about 700 words each. The features well contains three to five stories that range from 1500-6000 words each. These features tend to focus on analysis and reportage, but occasionally include photo essays, profiles, and other softer pieces. After the feature well comes the Fiction and Poetry section, consisting of a short story and several poems; lengths vary wildly from issue to issue. The Arts section contains a profile piece as well as several shorter articles and columns. Finally, there is a back page column.[28]

---

[26] Jessica Johnston (editor-in-chief, *This Magazine*), phone interview, January 21, 2007.

[27] Ibid.

[28] Ibid.

Initial planning begins several issues in advance. The editor-in-chief's ideal is to plan six issues ahead, but often only has plans for two to four issues after the current production cycle. *This Magazine* only recently returned to doing theme issues, and the process of deciding on issue themes is still relatively informal as a result. The editor-in-chief brainstorms and discusses possible themes with various members of the editorial staff. Once the theme is set, the editor-in-chief decides on the mix of features. Much closer to the beginning of an issue's cycle, the editor-in-chief sits down with each section editor and goes over possible articles for each section. The planning process is informal, in spite of attempts to introduce structure. The continuing lack of structure can be attributed to a deficit of time and resources. The ad hoc nature of the planning process is a common thread woven through *This Magazine*'s operations.[29]

Articles come in Word format via e-mail. Depending on whether the piece belongs in a particular section or in the feature well, one of the section editors or the editor-in-chief will perform the substantive edits and communicate with the author. The handling editor marks substantive edits in the text in Word and sends the document back to the author. Each article passes through several draft stages; in the case of non-feature articles, the editor-in-chief acts in a supplementary advisor role when necessary. Once substantive edits are finished, the handling editor passes the article off to the interns for fact checking. Short fiction and poetry currently skip over the fact-checking stage, though this may change in the future.[30]

After fact checking, the interns return the piece to the editor-in-chief for a read-through. The editor-in-chief in turn sends the piece to a copyeditor, who makes edits in Word and returns the article to the editor-in-chief. Then the article goes to the art director for layout. The next time anyone sees the story is together with the rest of the

[29] Ibid.

[30] Ibid.

issue, in the form of proofs. Proofreaders take pages for a weekend and send back changes on Monday; then the office staff proofreads the issue several times; then finally the editor-in-chief gives the issue one last proofread before it goes to the printers.[31]

The editor-in-chief tracks all articles during the process by way of an Excel spreadsheet and by making notes at the top of all articles indicating its status. This system works largely because the editor serves as a hub for all articles; all pieces return to the editor between workflow stages. Other people in the process generally do not know the whereabouts of every article, and simply know that when they receive possession of an article, it is their turn to edit the piece. Because the system the editor-in-chief uses to track articles depends on her ability to understand the markings and notations she uses, she occasionally becomes confused. The editor-in-chief used to simply note the status of an article by using shorthand notation in the filename of an article, but occasionally found herself wondering if a file marked for a particular stage meant the article needed to enter that stage or if that stage had already been completed. She noted similar problems with the Excel spreadsheet she uses.[32]

Despite the relative lack of structure to planning and, to a lesser extent, the production cycle itself, the editorial process works well. The workflow is routine and predictable, and the magazine never misses press deadlines. The pace of the process is relaxed enough that the editor-in-chief can delegate tasks to section editors without having to micromanage the process, and deal with unexpected situations without pushing the schedule back. A Magazines Canada travelling consultant noted no major deficiencies during a recent visit to the magazine's offices. One problem with the workflow is that it relies on the editor-in-chief to act as a point person for everyone else,

[31] Ibid.

[32] Ibid.

and coordinating the entire editorial process while also handling administrative duties for the magazine is a lot to deal with.[33]

The editor-in-chief did feel that despite the current workflow's success, there was room for improvement, which could include the use of software. The magazine has not done any research on workflow software because there has never been any need, and more pressing tasks have always come up. The magazine once used web forum software several years ago as a coordination and communication tool (since abandoned for lack of use), and also has a Basecamp subscription that was currently going largely unused for print operations. Originally set up for coordinating website operations, its use for editorial work on the print side has been limited; the editor-in-chief, for example, felt it was difficult to remember to check Basecamp for important notes and messages.[34]

While the magazine is open to the adoption of software in an abstract sense, the editor-in-chief was unable to come up with specific areas where she felt software would improve the process. In explaining the issue, she made an analogy to a software tool called Scrivener, a project management application aimed at writers. Until she began using Scrivener, she did not know she needed its functionality, but within a week it had made a significant improvement on her old writing process in ways she was unable to explain to her own satisfaction. The same challenge exists for the adoption of workflow software; it offered the potential to improve the workflow, but for the magazine, exactly where those improvements would be made was hard to predict.[35]

Shameless

- **Location:** Toronto, Ontario
- **Schedule:** three times per year

[33] Ibid.
[34] Ibid.
[35] Ibid.

- **Circulation:** 3,000
- **Audience:** teen girls and young women
- **Content:** politically-minded, feminist take on teen life and issues, covering arts, culture, politics and social issues
- **Respondent:** Megan Griffith-Greene, editor-in-chief

Shameless sits at the lower end of the magazine resources scale compared to the other magazines. The magazine's core editorial staff of four people works on a volunteer basis, and has no central office. *Shameless*'s editorial workflow is linear, with workload split fairly evenly between the three editors, and is managed manually without the use of software. Though the editorial workflow is adequate in that it produces a high-quality product with a relatively small volunteer staff, more rigour and organization would be a welcome enhancement. *Shameless* had not done any research into software but would consider it in the future.[36]

Four people make up the print magazine's core editorial staff: the editor-in-chief, two section editors handling features and reviews, and the art director. The magazine also has three to seven copyeditors and proofreaders working on any given issue. *Shameless* has no central office, so everyone works from home. All the staff members, including the editor-in-chief, are unpaid volunteers. An average of 30 writers and artists contribute to each issue. Submissions come from a variety of sources; the magazine maintains a slush pile, though the magazine rarely uses unsolicited submissions. The bulk of an issue comes from pitches and assignments. The magazine's art direction relies heavily on illustrations, which are generally assigned or created by the art director.[37]

An average issue contains about 50 articles, though contributors occasionally have more than one piece in an issue and editors write many of the smaller pieces themselves. A letter from the editor and a letters section make up the front-of-book. The

[36] Megan Griffith-Greene (editor-in-chief, *Shameless*), e-mailed questionnaire, January 19, 2008.
[37] Ibid.

columns and departments section follows, containing several pages of short pieces around 300-500 words each, and three to four columns of about 700 words each. The feature well contains three or four articles of about 1200-2000 words each, plus sidebars of 300-400 words. The arts section contains a mix of articles ranging from 400-1000 words, and the review section contains about twenty capsule reviews of about 150 words each. Finally, there is a back-page column of 500-600 words.[38]

Planning for an issue happens after the last issue's production cycle has ended. The three editors meet to discuss possible story ideas for the next issue based on pitches and queries they have received from writers and looking at topics they have covered in the recent past. The decision-making process is democratic.[39]

Documenting the magazine's workflow is difficult because there are no guidelines designed to enforce a specific way of working, and each editor tends to develop their own individual methods. Generally speaking, however, each article goes through roughly the same process. Articles arrive at the magazine as Word documents via e-mail. The various section editors act as handling editors for articles in their relevant section, with the editor-in-chief assisting with major problems like an article that has shifted considerably from the original specifications. Otherwise, the handling editor trades drafts with the writer of the piece for approximately a month. During this time, the editor-in-chief also reviews articles and makes notes if necessary.[40]

After the substantive edits, the article goes to copyediting and fact checking. The art director also receives an initial draft for layout purposes, and is notified of changes yet to be made. Drafts are e-mailed to the copyeditor, who makes edits on paper, merges them into the Word document and e-mails the revised draft back to the editor-in-chief.

---

[38] Ibid.

[39] Ibid.

[40] Ibid.

After the copyediting and fact checking stage is done, the piece is fitted into the layout, which can involve further cuts to the article's length. Depending on how far behind the article is in the editing process, it may enter layout before copyediting changes are complete, requiring changes to be merged into the InDesign document instead of the original Word document. Finally, once the layouts are finished, the magazine goes to a proofreading stage. From copyediting to final proofing changes, the process takes roughly another month.[41]

At the end of every major stage of editing, the article returns to the editor-in-chief. Thus the editor-in-chief acts as a coordinator for the entire process, and tracks the progress of all articles in a given issue. Other editors know they need to work on an article solely by whether the editor-in-chief has given them possession of an article. The editorial process is fairly linear; rarely is there more than one person working on a piece at any given time.[42]

Shameless is coping with the challenges of an all-volunteer staff fairly well, but the editor-in-chief believed adding structure and organization to the magazine's workflow would make the production process much easier. The major workflow challenges have to do with scheduling and organization. Everyone on the magazine's staff holds down jobs separate from *Shameless*, which means the simple act of coordinating meetings becomes far more difficult—a problem that also appears during the editing process itself. The extra time and effort spent on coordination and time management affects the quality of editorial work. In addition, if editors are especially pressed for time due to other commitments, they may not be able to give very much attention to articles later in the editing process.[43]

---

[41] Ibid.

[42] Ibid.

[43] Ibid.

Shameless's editorial structure changed considerably in the spring of 2007, with the magazine's original two editors retiring from the magazine's production. They were replaced by a single editor-in-chief and several section editors, positions that had not previously existed. As a result, the magazine has only had about half a year to come up with a new workflow tailored to the additional staff. A more rigorous workflow system is a priority for *Shameless*, according to the editor-in-chief.[44]

Shameless does not currently utilize a workflow management software package of any sort. The personal computers of everyone on staff makes up the magazine's computer infrastructure, so any workflow software would have to work under a variety of operating systems and environments. The magazine has not done any research on workflow software, but the editor-in-chief believes it is an obvious avenue for further investigation. She had several feature requests in mind when asked to discuss how software could improve the editorial workflow:

> I think it would be fantastic if there was a way that all editors could log in, update the status of their pieces, write any notes that other editors need to know about, and even upload drafts. [45]

Despite the challenges involved in setting up workflow management software, the editor-in-chief plans to look into possible software options, thanks in part to the issues she considered in responding to the study.[46]

Spacing

- **Location:** Toronto, Ontario
- **Schedule:** three issues per year
- **Circulation:** 5,000

[44] Ibid.

[45] Ibid.

[46] Ibid.

- **Audience:** younger Toronto residents with an interest in the city's urban issues and urban planning in general, especially urban planners, architects and students
- **Content:** news, features and essays about the urban makeup of Canadian cities, especially Toronto
- **Respondent:** Dale Duncan, managing editor

Spacing is a bit of an outlier with regards to its staff composition and workflow. The core staff is eight people, seven of which are editors with fairly equal responsibilities and duties; the two most senior editors work out of the central office on a significant basis, while the other members use the office space occasionally. The whole staff works on a volunteer basis. *Spacing*'s editorial workflow is linear, but differs from the other magazines in that six of the editors (minus the arts editor) take on a portion of an issue's editing workload on a rotating basis, such that each editor will perform first edits on one-sixth of the content, and second edits on a different sixth of the content. *Spacing* is one of the few magazines to utilize any form of software to help manage the workflow, in this case discussion board software. The editorial process works well, with the discussion board playing a critical role in staff communication.[47]

Spacing's editorial staff consists of seven editors and a copyeditor. Dale Duncan is the magazine's managing editor. The magazine's publisher acts as a lead editor and art director during the editorial cycle. The publisher and managing editor are the only two members on staff who spend a significant amount of time in the magazine's office space. The four associate editors and the arts editor all work outside the office on a part-time basis, as does the copyeditor. The magazine uses approximately 40-50 contributors per issue, three-fifths of them being writers and the difference split between photographers and illustrators. *Spacing* also utilizes five interns, though only one is full-time and their

[47] Dale Duncan (managing editor, *Spacing*), phone interview, November 22, 2007.

duties are often split between writing articles for the print magazine, handling the magazine's website, and coordinating events.[48]

A typical issue of *Spacing* consists of four main sections. Curious City contains an average of six recurring features of no more than a page, plus a number of short news capsules (usually around four or five). Inner Space is the magazine's columns and essays section, and often consists of four columns of a page each, though the section fluctuates in number and size. The feature well is where the issue's theme is enforced, and averages about fifteen articles an issue. Three of the features will extend over 2,000 words in length, and three more over 1,000 words. The rest of the features are approximately a page in length. The stories tend to connect to one another, and are often accompanied by up to five sidebar pieces per issue. Finally, the Backspace section contains four Outer Space stories about other cities, a reviews section containing an essay about the theme and several capsule reviews, and a back page column.[49]

Every issue of *Spacing* has a theme that is usually determined by the entire staff sometime before the previous issue has gone to press. Planning for future issues can extend up to three issues in advance, but generally *Spacing* only has the current issue in production and the next issue planned. Most of the magazine's content comes from a submissions process. The magazine puts out a call for pitches based on the theme, and the editors pick a number of suitable pitches from submissions. Stories are also assigned to cover aspects of the issue's theme that the accepted submissions do not address. The staff keeps track of submissions via an electronic discussion board system called phpBB. All pitches are posted to the board for review by all the editors and discussed during one of the magazine's biweekly meetings in the office space. The board allows editors to

[48] Ibid.

[49] Ibid.

comment on pitches beforehand, but decisions on which pitches to pursue are made during the meetings by the core editorial team.[50]

Substantive editing duties for the entire issue, aside from the reviews section, are split relatively evenly between the four associate editors, the managing editor and the publisher. The reviews section is handled by the arts editor. For the rest of the issue, every article is assigned two editors—a handling editor for the first edit, and a second editor for additional input. Three editors handle the feature well, two editors take on front of book material, and one editor handles the Backspace section; these assignments rotate every issue.[51]

Drafts arrive at the magazine in Word format via e-mail. Each article gets its own thread on the discussion board so that everyone on the editorial staff can review the article. The handling editor creates a thread by copying the contents of the article out of the Word document and pasting it into the first post of the thread. All subsequent drafts are also posted to the board as they come in from the writer. Edits are marked electronically in Word, though it is generally left to editors whether they wish to use the Track Changes functionality or simply insert inline notes to indicate changes.[52]

Once the handling editor is satisfied with the state of an article, the piece goes to the second editor for another round of edits. At this point edits are generally minor tweaks; if major issues crop up in the second edit, the piece goes back to the writer via the handling editor. As with the first round of edits, drafts of the article are posted to the discussion board. During both editing stages, other editors may leave comments on the

[50] Ibid.

[51] Ibid.

[52] Ibid.

piece in the article's discussion board thread, though generally this is done only for the benefit of the handling editor, and isn't actually a part of the editing process.[53]

After both editors have finished making substantive changes, the piece goes to the copyeditor, who posts a copyedited version of the article on the discussion board. The publisher, now acting as art director, then takes all copyedited drafts and exports them into layouts in InDesign. The art director presents preliminary layouts during an editorial meeting for feedback, and then prints out a round of layouts for the editors to perform a second copyedit on paper. The art director and managing editor input all changes from the second copyedit and print out a final proof for all the editors to review.[54]

Spacing editors track articles in the editorial cycle through the electronic discussion board. Each article has its own thread in which pertinent information like handling editors and the article's status is recorded. The board also serves as the magazine's de-facto file server, as each article thread also holds the text of every draft during the process. The board only tracks articles up to the layout stage, however; the staff was unable to post InDesign documents or PDFs to the discussion board, meaning editors had no way to review or track layouts electronically.[55] The *Spacing* staff also keeps track of the issue's progress through the editorial meetings, which become weekly meetings in the late stages of editing and production. These meetings serve as the main method of tracking the issue's status once articles enter layout.[56]

The magazine's current workflow system is quite good at meeting the particular challenges the magazine faces. With an all-volunteer staff working mostly outside the

[53] Ibid.

[54] Ibid.

[55] phpBB actually does allow users to attach files to comments and topics. However, this functionality may not be obvious to novice users and additionally may require changes to phpBB's configuration or user permissions to activate the feature.

[56] Duncan, phone interview.

office, communication and coordination are crucial issues that the electronic discussion board goes a long way towards solving. By having all documents and some of the editorial discussion on the board, the staff has a single source of information to rely on. In contrast to most of the magazines that took part in the study, the various members of the editorial staff share duties fairly equally. Most importantly, there is no central hub person who controls and tracks the magazine's workflow, while the rest of the staff sees only their relevant portion of the workflow. Through the discussion board, all the editors can ascertain the status of any article in the issue relatively easily.[57]

In terms of problem areas, the current system occasionally leads to confusion and stress, partially because the editorial staff works largely as a collective rather than as separate individuals, with relatively little enforcement of hierarchy—something the managing editor believes is both an asset and occasionally a hindrance. She also described the problems that seem to be inherent to an all-volunteer staff: not enough time to do too much work, including non-editorial work that often gets done by editorial staff out of necessity (the managing editor, for example, is also the subscriptions manager). One improvement the managing editor put forth is to have someone who could look at the production of an issue from a big picture standpoint and see the magazine as a whole. Overall, however, she feels the current system is quite good at meeting the magazine's needs.[58]

Spacing adopted the phpBB internet discussion board software as a communications hub very early in the magazine's life, and has made the board a central part of the magazine's workflow. The discussion board serves two major purposes for the magazine: as a file server and as a way to discuss editorial concerns between meetings. Since *Spacing* lacks a robust IT infrastructure and has no central file server in the

[57] Ibid.

[58] Ibid.

traditional sense, the discussion board's ability to store the magazine's articles during production is invaluable. The discussion board's file storage capabilities, as the magazine currently uses them, are not exactly ideal. As with layouts, editors do not attach Word documents to article threads directly, but rather copy text out of the documents and paste it directly into a discussion board post.[59] This behaviour can cause a number of problems when it comes to moving that text into InDesign layouts. Web-based discussion boards were never designed to prepare text for Word or InDesign, and so basic features like text formatting have to be redone every time an article enters or leaves the discussion board.[60]

The discussion board's hierarchical discussion structure—forums for each issue containing comment threads for each individual story—also helps to structure editorial discussion. However, most editorial discussion occurs during the in-person meetings rather than on the board, partially because the same organizing structure that makes it easy to add comments to specific articles also makes it hard for editors to see at a glance what is new on the board. Important notes can get lost if an editor does not keep a close eye on all the article threads, and as comment volume on the board rises, it becomes more difficult to find comments that might be relevant to an editor's workload. As a result, the staff still relies on editorial meetings to highlight urgent issues. One feature the managing editor proposed to mitigate the problem is a feature that would alert editors to new comments on specific posts.[61]

Since adopting phpBB, *Spacing* has not looked further into workflow management software. This is partially because of the inability to devote time or money

---

[59] See note 55 for an explanation of why *Spacing* does not use attachments.

[60] Duncan, phone interview.

[61] Duncan, phone interview. A note about thread alerts: phpBB does, in fact, have this functionality; the Subscribe to Topic feature allows users to track comments made on specific threads. However, as with the file attachment feature, this functionality may not be obvious to novice users and may also require changes to phpBB's configuration or user permissions.

to the issue, and partially because the current system is workable. However, the managing editor noted obvious areas for improvement, and additionally said it would be ideal if the board could be adapted to handle layout files as well, so that staff could more easily discuss changes or improvements without having to wait for the editorial meetings. *Spacing* would be very interested in any new system that was workable and affordable.[62]

Briarpatch

- **Location:** Regina, Saskatchewan
- **Schedule:** eight issues per year
- **Circulation:** 2,500
- **Audience:** left-leaning, politically-minded Canadians; when launched in 1973, originally aimed at unemployed and low-income readers
- **Content:** alternative viewpoints on Canadian politics leaning towards the progressive end of the spectrum
- **Respondent:** Dave Mitchell, editor-in-chief/art director/designer

Briarpatch resembles a smaller version of *This Magazine* in some ways. The core editorial staff consists of nine people, most of whom work on a volunteer basis. *Briarpatch* has a central office that some, but not all, of the staff uses. The editorial workflow is linear, with most articles falling under the editor-in-chief's purview, and associate editors acting as handlers for the rest. *Briarpatch* uses Basecamp, a web-based project management application, to help coordinate its workflow. Mitchell said that while the magazine has only recently adopted the application, it has already seen substantial improvements over the old manual process of managing the workflow.[63]

---

[62] Ibid.

[63] Dave Mitchell (editor-in-chief, *Briarpatch*), e-mailed questionnaire, December 12, 2007; Dave Mitchell, supplementary e-mail correspondence, July 16, 2008.

The magazine's editorial staff is largely volunteer-driven. The exceptions are the editor-in-chief, who also serves as the art director and designer, and the publisher, whose role is largely non-editorial but does involve some proofreading and display writing. The rest of the staff includes four associate editors, three copyeditors and one proofreader; many of these people live and work across Canada. *Briarpatch* has a Regina office that the editor-in-chief and publisher both use, but all other communication with staff occurs electronically or over the phone. The average issue also includes the contributions of eight to ten writers and three to five illustrators and photographers.[64]

Briarpatch's editorial mix is fairly simple, and can be neatly divided into features and departments. Each issue contains roughly five to seven features, ranging in length from 1,200-3,000 words each. In the Departments section, there is an editor's column, reader letters, one to three reviews of about 500 words each, and a Parting Shots column of about 700 words.[65]

Roughly half of the magazine's eight issues are devoted to specific topics. For themed issues, the editor-in-chief selects articles from a call for submissions made at least three to four months prior to the issue's publication. The slate of articles for unthemed issues is filled largely from accumulated queries in the months before publication. In both cases, a number of articles are also assigned to writers, usually past *Briarpatch* contributors; generally this number is higher for unthemed issues. The editor-in-chief goes through all submissions and makes decisions on what to publish based on a submission's merit, the relationship with the writer, and the timeliness of the article. Submissions are tracked on paper; the editor maintains a file folder containing all queries and responds to all submissions within six weeks.[66]

---

[64] Mitchell, e-mailed questionnaire.

[65] Ibid.

[66] Ibid.

Initial drafts are submitted to the magazine in Word format via e-mail. Both the editor-in-chief and the associate editors handle substantive editing duties, with the editor-in-chief handling the majority of an issue's articles and the associate editors splitting the remainder. In all cases, the editor-in-chief reads the piece and, if the article is assigned to another editor, will offer brief notes about how to rework the article if necessary. The handling editor edits the electronic document using either Word or OpenOffice, an open-source equivalent to Word. The Track Changes functionality is used to mark suggested edits and notes in the document. After the handling editor and author trade several drafts and the article is deemed fit, the article moves to the copyediting stage. The art director will also take the current rough draft and lay it out in InDesign for rough fitting.[67]

At this point, the article is uploaded to Basecamp. The article is copied out of the Word document and pasted into a Basecamp writeboard for further editing. Writeboards are essentially web-based documents that allow collaborative editing and track edits and revisions. During the copyediting stage, several editors may be involved in the process: the editor-in-chief, an associate editor, and the copyeditor. Each person will copyedit the document and make changes in turn. Generally the handling editor is not involved in copyediting, though it does happen occasionally. Should major changes be required at this stage, the handling editor will send the article back to the writer for additional input.[68]

Once copyediting is complete, the editor-in-chief, now acting as the art director, takes the article out of Basecamp and imports the text into a layout in InDesign. After the pages have been designed, the art director creates PDF proofs and sends them to the proofreader and the publisher for proofreading. The proofs are also uploaded to

[67] Ibid.

[68] Ibid.

Basecamp as files (not as writeboards) for the associate editors and copyeditor to review, though the bulk of the proofreading changes come from the proofreader and publisher. After the proofreading changes are merged with the InDesign document, the issue is ready for press.[69]

The editorial staff tracks the status of articles in the editorial process in two ways. For each issue, the editor creates a Basecamp project representing the issue, and starts a writeboard called "article line-up" that all editors can edit. Each article is listed in the writeboard, along with the handling editor and the article's status. The editorial staff also keeps track of articles by way of notes inserted at the top of all article whiteboards with some brief comments about the article's status. Copyeditors are assigned to pieces in a separate writeboard that is similar to the article line-up but lists the copyeditors assigned to the articles (or open spots if articles still require copyediting volunteers).[70]

The magazine started using Basecamp in September 2007, very close to the time the editor-in-chief sent in his responses for this study, and he was able to compare the current workflow to the process the magazine previously used. Basecamp allows the magazine to better utilize the other members of the editorial staff, as much more of the process is now exposed to the other editors. As a result, the other editors are able to contribute in a more meaningful way—for example, associate editors are now more aware of all the articles slated for an issue, and are better able to volunteer to take handling or copyediting duties on articles that are particularly interesting to them.[71]

Basecamp is not without its deficiencies, however. Writeboards are ill equipped to handle formatting when text is pasted in from Word. That means all of an article's formatting has to be marked manually in the text using editor's marks (such as _italics_

[69] Ibid.

[70] Ibid.

[71] Ibid.

and *bolding*), though copying the text out of the writeboard and into InDesign was relatively trouble-free.[72] The editor-in-chief also noted that the magazine may not have had enough time to truly explore Basecamp's functionality at the time of the study, and so may be unfamiliar with other Basecamp quirks.[73]

However, despite those concerns, the magazine's Basecamp-centric workflow represented an improvement over the old workflow. Basecamp's feature set, internet-based functionality, ease of use and relatively low cost are all major factors in *Briarpatch*'s adoption of the software. The magazine had researched other workflow options before deciding on Basecamp. Among the programs and services researched were Google Documents, which was deemed too limited; various implementations of Adobe InCopy-based systems, which were too expensive and required a central server; and Bricolage, eventually dismissed as requiring too much technical ability to set up. Of all these choices, the editor-in-chief feels Basecamp is the best choice for the magazine's current situation.[74]

Comparing the magazines

One of the questions this study was designed to answer was how much variance is there in the way different magazines work. If all small magazines have roughly the same personnel, produce roughly the same types and amounts of content, and perform roughly the same tasks in the same order, then coming up with a set of requirements for workflow software aimed at small magazines should be relatively simple. If, on the other hand, every small magazine has their own way of doing things that contrast wildly with

[72] InDesign is able to capture formatting from Basecamp writeboards largely because of the confluence of features built into Basecamp, Internet Explorer and InDesign. Basecamp was never explicitly designed to handle InDesign export, nor was InDesign explicitly designed to handle input from Basecamp.

[73] Mitchell, e-mailed questionnaire.

[74] Ibid.

other magazines, then imagining a software package that meets the needs of most publications will be much more difficult.

The seven magazines that responded to the study share certain situational aspects. Most of them publish largely or completely non-fiction content, as opposed to poetry or prose fiction; all have small editorial staffs under 20 people; most rely heavily on volunteer or part-time labour who often do not work out of the magazine's office space. Certain magazine types may be underrepresented in this study, such as literary journals and small magazines staffed mostly or completely with paid workers. However, the magazines in the study are owned by various types of publishers, based in different geographical locations, focused on a variety of subjects, and targeting audiences of different types and sizes.

Perhaps because of the traits the various magazines share, the workflow models of every magazine are more similar to one another than different. Though some magazines may concentrate more on certain stages of an issue's production, or cut out stages that other magazines include in their process, the basic workflow is the same:

- For the magazines that asked for or accepted submissions, a review stage where articles would be accepted or rejected for publication, often protracted and performed by only the senior editors or editor-in-chief;
- A substantive editing stage, where a handling editor would collaborate with the article's author and make large-scale changes to an article's structure and meaning;
- A copyediting stage, where one or more copyeditors would edit an article for grammar, punctuation, spelling and house style;
- A layout stage, where an art director or designer would place an article into the layout of the magazine;
- A proofreading stage, where proofreaders or other members of the editorial staff would review galleys to catch minor mistakes before the final output was sent to the printers.

The process in general is best characterized as a linear progression much like an assembly line, where an article is passed from stage to stage and editor to editor in a straightforward fashion. Generally only one person in the workflow has possession of an article at any given moment, and articles tend to move forward in the workflow—that is, once a piece has passed the substantive editing stage, it does not generally return to that stage later in the workflow. In practice, this linear model has several implications. First, staff members often have a limited set of duties and a narrow perspective on the magazine's editorial and production process, restricted largely to the specific workload of each staffer. It also means one person, usually the editor-in-chief, oversees the entire editorial process. That person is responsible for knowing where every article is in the process, and who is assigned to various editing roles on each article. The editor-in-chief furthermore acts as a hub for the magazine's editorial operations, both in the sense that the editor-in-chief tracks the status of articles, and in the sense that articles literally return to the editor-in-chief's possession before being sent to the next stage.

Though all the magazines rely on the same basic progression, there are deviations from the norm. First, there is the issue of how technology influences each magazine's workflow. *Briarpatch* and *Spacing* utilize software to help manage their workflow and facilitate communication between staff members, whereas the other magazines rely largely on a mix of ad hoc tracking strategies like shorthand notes in filenames and keeping track of articles on paper or using an Excel spreadsheet. *Briarpatch* and *Spacing* also have more complex workflows than the other magazines. *Spacing*'s model is the most complex, with six editors share substantive editing duties in such a manner that has two editors reading each article, and each editor handling a different section of the magazine from issue to issue. Such a system requires significantly more coordination than one where only a single editor handles substantive editing duties, where there are relatively few substantive editors, and where each editor maintains ownership of a single

section from issue to issue. *Briarpatch*'s process is more similar to the other magazines, but does involve a copyediting stage where multiple people have ownership of an article during the stage. Editor Dave Mitchell noted that because the software the magazine used, Basecamp, was able to show the rest of the staff who is assigned to articles and what stage each article is at, it requires less coordination on his part.[75]

Other differences are more minor and seem to be more the result of institutional inertia—in other words, "the way it's always been done." For example, *This Magazine* explicitly includes a factchecking stage in their workflow[76], something none of the other magazines do (though several magazines indicated that they use factcheckers as well). The *Queen's Alumni Review* combines several tasks into some stages; for example, their copyeditor is also a proofreader and factchecker, and the editor emerita also acts as a proofreader of sorts.[77] Many other magazines indicated that their proofreading stages are somewhat unstructured; *Up Here* uses staff from its sibling magazines for proofreaders, while *Spacing* and *Shameless* have some or all of their regular editorial staff proofread galleys.[78]

One thing every magazine has in common was their art workflow. With one exception, the art workflow is handled by one person, the art director. The art director is responsible for commissioning and editing photography and illustrations, and for the most part the art director handles everything related to the magazine's graphics and layout. Interactions with the editorial staff usually happen near the beginning of the production cycle, when art and editorial would come up with graphic treatments for each story. Depending on the magazine, art directors also pass on illustrations and photos to various editorial staff (usually the editor-in-chief) for feedback. The single major

[75] Mitchell, e-mailed questionnaire.

[76] Johnston, phone interview.

[77] Cuthbertson, e-mailed questionnaire.

[78] Spitzer, phone interview; Duncan, phone interview; Griffith-Greene, e-mailed questionnaire.

exception to this trend is *Up Here*, who also has a photo editor/researcher. The photo editor and art director split art commissioning and editing duties into photography and illustrations. The art director also makes final picks from the photography the photo editor provides.[79]

None of the editors of the magazines seem to worry very much about the art workflow, though this may be partially because two of the magazines, *Briarpatch* and *Spacing*, have editors that double as art directors. Most of the other magazine editors in the survey express a mixture of trust in their art directors and an admission that art was not their thing, leading to a general hands-off approach. The layout process tends to be more of a collaboration between the editor-in-chief and the art director, though these interactions are more informal and ad hoc. The level of interaction between art and editorial at the layout stage also varies more between magazines during the layout stage than in the art editing stage.

Workflow satisfaction

Though every magazine has a workflow that works well enough to produce magazines of sufficient quality on a regular basis, the level of satisfaction each magazine has with its workflow varies significantly. *Taddle Creek* and the *Queen's Alumni Review* both rate their workflow processes highly, and see no reason to change their workflows at all. Both magazines have been using the same workflow system for years; Conan Tobias of *Taddle Creek* said its system has "worked perfectly for us for 10 years,"[80] while Ken Cuthbertson of the *Queen's Alumni Review* believes that after "many years of trial and error," the staff has become comfortable with the system, and said its workflow is

[79] Spitzer, phone interview.

[80] Tobias, e-mailed questionnaire.

"lean, mean, and efficient with a minimum of fuss."[81] Not surprisingly, both magazines also believe that adopting any sort of workflow software will be either pointless or possibly even counterproductive. Cuthbertson states that adopting workflow software would be additional work for very little benefit. He offers in his questionnaire these final thoughts on workflow software:

> Being a veteran of twenty-plus years in the magazine industry (at Maclean-Hunter—now Rogers—and at Queen's) and with another decade in newspapers, I think I have a pretty good handle on the production process. My sense is that workflow software would work best in a large shop where there are a lot of people involved in the production process and the pace is quick (i.e. at *Maclean's*, *Chatelaine*, *Reader's Digest*). It could also be of help at publications where there is a high staff turnover or the staff are new to the business. To be candid, I don't see how workflow software would be of any use to me or to the editors at other publications where the staff is small and the deadlines spread out. Like so (too!) many things nowadays, more software to buy, install, master, and use would complicate the production process as much as it might help make it more efficient. I could be wrong about that, but that's my take on the situation.[82]

At the opposite end of the spectrum lie *Shameless* and *Up Here*, who both believe their workflows are adequate but fragile. *Shameless* in particular claims the need for substantial improvement. Megan Griffith-Greene, the editor of *Shameless*, said that while the magazine is doing well considering the volume of work and the all-volunteer staff, *Shameless* has "to work on a more regular workflow to keep everyone feeling connected to the magazine."[83] Aaron Spitzer, the editor of *Up Here*, feels the magazine's current system works, but that certain aspects of it could use more structure, like the planning process for issues and the proofreading stage. He also noted that because most of the conversations about the state of the current issue in production happen largely

[81] Cuthbertson, e-mailed questionnaire.

[82] Cuthbertson, e-mailed questionnaire.

[83] Griffith-Greene, e-mailed questionnaire.

between himself and the senior editor, other people on staff tend to feel slightly disconnected from the process.[84]

In the middle are *This Magazine* and *Spacing*. The editors of both magazines express satisfaction with their current workflow systems, while remaining open to the possibility of further improvements. Jessica Johnston of *This Magazine* feels their workflow is "routine and predictable for the most part; everyone knows how it works; things happen at the same time every cycle; there are no surprises." Though Johnston did say the task of keeping the editorial process on track and staying organized is a challenge, she feels that's only natural "considering I'm the point person and everyone works volunteer out of the office." Johnston has no major issues with the magazine's workflow, though she sees the potential for improvement in an abstract sense. [85] Dale Duncan of *Spacing*, by contrast, has several suggestions for areas of improvement. Overall, Duncan feels the system the magazine has devised is quite capable, but that despite the workflow's comprehensive structure, "even so it feels like we're not organized. Sometimes I feel like it's rushed." She also believes that technological improvements like a central file server could improve the current workflow by allowing editors to see layouts in advance of the weekly editorial meetings—a concrete suggestion with obvious results. [86] Interestingly, both editors have very similar comments about their particular role in the workflow. Johnston and Duncan both say they had significant non-editorial duties that took time away from editorial work. They both feel the editorial process would be much smoother if not for the additional workload.[87]

---

[84] Spitzer, phone interview.

[85] Johnston, phone interview.

[86] Duncan, phone interview.

[87] Johnston, phone interview; Duncan, phone interview.

Briarpatch is a special case because the magazine only recently adopted a new workflow system based on Basecamp. Dave Mitchell, the editor of *Briarpatch*, briefly compares his experience with both workflows:

> Our current process is a big improvement on what we did before, which basically involved me doing most of the work, but bouncing out articles at various stages in the editing process to volunteer editors for their input and reworking. I found that process very difficult to manage effectively, since I was effectively the hub, with very limited ability to broaden editorial participation from volunteers. It also had the disadvantage that if I was tied up with one particular article, or some entirely other matter, everything else was on hold until I could manage to pass the other articles along to the next stage. Our current process, on the other hand, has the advantage of centralizing the "bank" of articles somewhere that all of the editors can access and provide feedback on them. This has improved our ability to match editors with articles of interest to them, since editors will often keep an eye on the line-up as it's developing and volunteer to serve as handling editor or copy editor of articles of particular interest to them.[88]

Several months after submitting his questionnaire, Mitchell said he has not changed his opinion of Basecamp after becoming more familiar with the software, and said it still performs the tasks the magazine needs the software to do.[89]

Room for improvement?

While several of the magazines surveyed—the smaller ones most notably—feel quite satisfied with their workflow, the majority of magazines see the potential for improvement, even if they feel their workflow meets their current needs. Many of the problems mentioned have to do with the issue of organization and process management; things like tracking the status of articles, the workloads of individual editors, and keeping everyone on deadline are often the responsibility of a single person. This,

[88] Mitchell, e-mailed questionnaire.

[89] Mitchell, supplementary e-mail correspondence.

combined with the relative lack of technological ability of most editors, means the habits and practices magazines often resort to in order to manage the editorial workflow are developed on the fly, require a fair amount of attention and maintenance, and are occasionally unreliable. *Up Here*'s ad hoc filename conventions for recording the status of articles or *This Magazine*'s coded status spreadsheet file are examples of solutions that each editorial staff came to naturally, but lack a certain rigour.

A wide array of software products exist that can solve some of these problems, however, by offloading the time and effort spent on manually tracking the status of articles and editors onto a computerized system. Moreover, because the workflow models of the seven magazines are fairly similar, any workflow software package that works for one of them can work for all of them with minimal effort. Finding a solution that can assist a wide variety of small magazines, then, would merely involve finding a software package that works well with the common aspects of small magazine production.

The software needs to model a linear workflow model where individual editors take possession of articles one at a time in an assembly-line fashion, making modifications at each stage until the original draft becomes a finished product. It also has to deal with the reality that many small magazines do not have office space or centralized computer infrastructure, and thus the software needs to be able to handle editors communicating and interacting with one another remotely, often via the internet. Because editorial staffers often work on a volunteer or part-time basis, they have little extra time to devote to the magazine beyond their editing duties, so it is also crucial that the software be easy to learn and use. All these points should be kept in mind when considering the array of existing software packages available.

Chapter 3: A brief review of existing workflow solutions

Surveying the landscape

A major issue facing small magazines looking for workflow solutions is the complete lack of products targeting the small magazine market. Magazine industry-focused publications like *Folio* run many articles about adopting content management and workflow systems, but often in the context of much larger publications with bigger budgets and staffs.[90] As a result, insofar as small magazines perform any research into software at all, they tend to adopt software that was originally designed for other purposes. Of the magazines surveyed for this study, the most extreme case was *Spacing*'s use of web-based discussion board software as a communications and planning hub. *Briarpatch*'s attempt to find workflow software was the most involved search of all the magazines; some of the packages they looked at appear in this review of workflow solutions, as do software packages adopted by publications in other fields such as academic and web-based publishing.

Many magazines, however, did not make any substantial attempt to seek out workflow software. In fact, most of the magazines that took part in the study fell into two camps. Either they had not researched editorial workflow software but were curious to learn more, or else they had not researched editorial workflow software and were largely

[90] Examples include Marrecca Fiore, "Publishers Turn to Content Management Systems to Increase Productivity and Profitability," *Folio*, September 19, 2006, http://www.foliomag.com/2006/publishers-turn-content-management-systems-increase-productivity-and-profitability (accessed February 11, 2008); Tony Silber, "Switching Edit/Design Platforms," *Folio*, August 2, 2007, http://www.foliomag.com/2007/switching-edit-design-platforms (accessed July 17, 2008); and Matt Kinsman, "How Technology is Transforming Magazine Publishing," *Folio*, June 30, 2006, http://www.foliomag.com/2006/how-technology-transforming-magazine-publishing (accessed July 17, 2008).

dismissive of their usefulness. In both cases, of course, no research has been done. There are several possible reasons for this lack of attention. Adopting a new workflow, especially one that requires adoption of new software where previously there was none, may simply be too much to ask of small magazines, whose staffs tend to be overworked, underpaid, and relatively lacking in technological knowledge. This is especially true for magazines that appear to be coping with the current workflow, even if the potential for improvement is obvious to them. Another major issue is the complete lack of any software aimed specifically at a small magazine audience; the only software aimed at the magazine market at all is priced well out of reach of most small magazines. This can prematurely end a magazine's line of inquiry before it uncovers other software packages that may be more suitable.

This chapter will cover a representative sample of options available to small magazines looking for editorial workflow software. The options were culled largely from the research reports of the magazines participating in this study, as well as industry publications such as *Masthead* and *Folio*. This chapter is not intended to be an exhaustive survey, as for each category there can be a large number of competing packages. This is especially true for workflow and content management systems aimed at the web publishing market. Rather, this chapter is intended to illustrate the categories of software a small magazine is likely to encounter in its research, and the pros and cons of each category. I tested all the programs with specific sub-chapters (Smart Connection Pro, Bricolage, Open Journal Systems and Basecamp), either via a full-featured trial or the freely available product.

Generally speaking, the software options fall into four categories. The first, consisting of what I've called "enterprise-class integrated workflow solutions" for lack of a better identifying term, contains the workflow and content management systems

produced by publication-oriented developers like Quark and aimed at large magazine publishers. The second category is web-based content management systems (CMS), which may or may not have comprehensive workflow modules, and are designed for use largely with web publishing ventures. The third category is workflow systems aimed at an academic publishing audience. Finally, there are project management tools that are aimed at small businesses in general, and are not customized for any sort of publishing task.

Enterprise-class integrated workflow solutions

Used by newspapers, magazine publishers and corporate publishing arms, "enterprise-class integrated solutions" are theoretically an obvious choice for any publisher who wants to organize their workflow process and make it easier for staff to collaborate on publishing projects—obvious largely because there are no other workflow products aimed at a print magazine audience. Products that fall into this category include *Quark Publishing System* (*QPS*), Softcare's *K4* and *K2* systems, and WoodWing's *Smart Connection* series of products. There are many other turnkey and custom-build systems available, but all share certain fundamental traits. Such systems are "enterprise-class" in the sense that the target audience tends to be larger publishers, and "integrated" in the sense that such products usually offer a tightly integrated set of applications to handle all editing and layout duties (for example, QPS's QuarkXPress for layout and QuarkCopyDesk for copyediting).

Enterprise-class solutions tend to be designed for relatively large workgroup installations; QPS 7's system requirements list several configurations for varying

workgroup sizes ranging from 30 to 150 users.[91] As of February 2008, Softcare advertised on its website an install base of 256 workflow systems with an average of 55 seats per site.[92] Until relatively recently, enterprise-class solutions were also designed specifically for use on a local-area office network and integrated with Windows or Mac layout and copywriting applications. QPS is tightly integrated with the QuarkXPress layout application and the QuarkCopyDesk copywriting application, while K4 and Smart Connection are designed for Adobe's InDesign and InCopy applications. In the time since such systems were first developed, the internet has matured into a stable platform for working remotely; in response all three systems now boast internet-capable "thin clients" that allow remote users to perform editing and layout tasks outside the office. Nevertheless, the basic paradigm still revolves around the use of the layout and copyediting applications, either locally or over the web, to do most of the work.

Neither Quark nor Softcare quote standard prices for their software offerings any more, preferring to set prices based on the number of simultaneous users, or "seats," and several other factors like maintenance, training and service contracts. Nevertheless, it is possible to find estimated cost figures from published case studies. For example, *Folio* published an article about *The Washingtonian*'s switch from QPS to K4, a transition that occurred in January 2006. The final cost was pegged at $150,000 US for a K4-based system designed for 40 editorial and production staff.[93] In another article published in September 2006, *Folio* interviewed Mark Walter, the director of business development at Managing Editor Inc., a North American distributor for Softcare's K4 system. Walter said the least expensive product Managing Editor sold was approximately $25,000 US.[94]

---

[91] Quark, "Quark Publishing System 7 System Requirements," http://www.quark.com/products/enterprise/modules/qps/tech_info/sys_req.html (accessed February 11, 2008).

[92] Softcare home page, http://www.softcare.de/ (accessed February 11, 2008).

[93] Silber, "Switching Edit/Design Platforms."

[94] Fiore, "Publishers Turn to Content Management Systems to Increase Productivity and Profitability."

WoodWing's Smart Connection Pro software costs $1019 US per user, and comes with Adobe InCopy.[95] Though much less onerous than the $25,000 minimum price tag of K4, $1000 per user is still quite expensive for most small publications. Factor in additional costs for computer infrastructure and (in the cases of magazines without office space) hosting costs, and it's clear that enterprise-class solutions are out of reach for most small magazines.

None of the magazines interviewed for the study used an enterprise-class solution, and very few had even researched one; *Briarpatch* looked briefly into adopting one of the Adobe-based solutions but dismissed the idea because it was too expensive and asked too much of the magazine's limited computer resources.[96] This situation is unlikely to change in the future, mainly because of cost and resource considerations. Even so, a review of one of the enterprise-class solutions may offer insights into functionality that can be carried over to a theoretical low-cost system relatively easily.

WoodWing Smart Connection Pro: an enterprise-class integrated workflow solution

Smart Connection Pro is the entry-level workflow offering from WoodWing Software. Designed around Adobe InDesign and InCopy, Smart Connection Pro is touted as a workflow system that allows designers and editors to work independently of one another without having to worry about where the other person is in their workflow. Though it lacks numerous features found in more expensive and complex systems like QPS, Softcare's K4 or WoodWing's own Smart Connection Enterprise (such as the ability to interface with third-party applications like Word and the ability to access the system remotely via a web browser), there are also several advantages that small publishers

[95] WoodWing Software, "Wooding web shop," http://woodwing.com/en/WoodWing_Store (accessed February 11, 2008).

[96] Mitchell, e-mailed questionnaire.

might appreciate. Recent adopters of WoodWing's more advanced Smart Connection Enterprise include Express Newspapers and the New York Sun.[97]

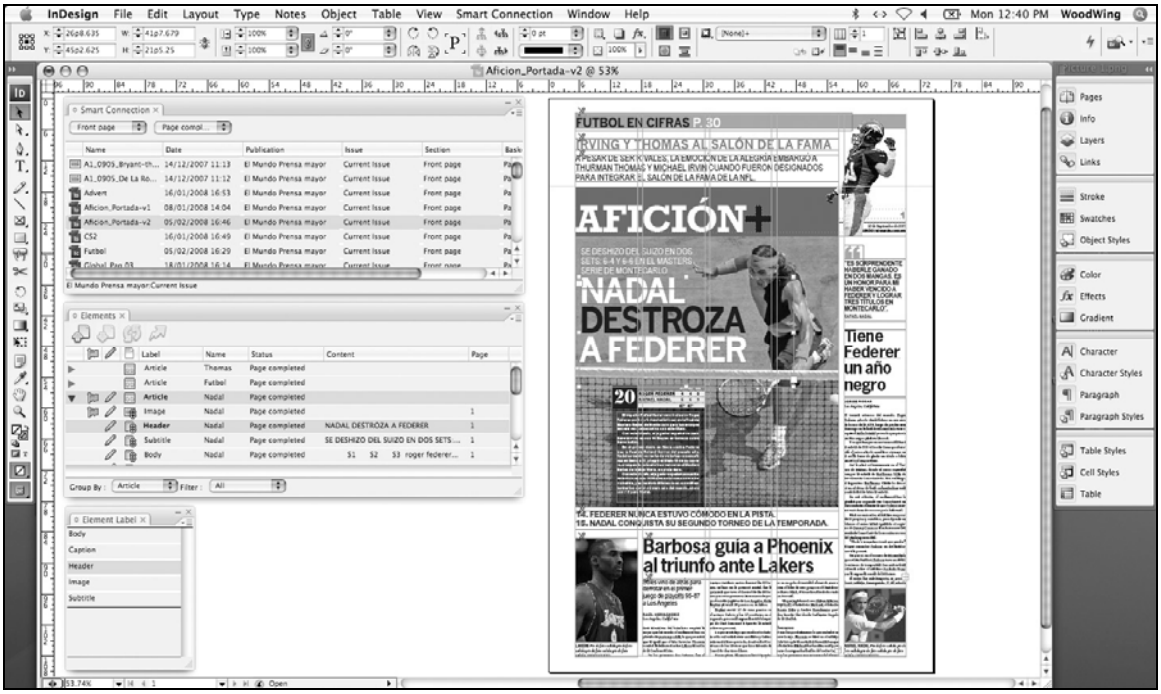

Figure 1. Smart Connection Pro interface in Adobe InDesign CS3. Note the two palettes on the left showing all articles in the system and articles checked out in InDesign. © WoodWing Software, by permission. (http://www.woodwing.com/en/Smart_Connection_Pro)

Smart Connection Pro: installation

One of the key differences between Smart Connection Pro and the more advanced packages is that Smart Connection Pro does not require server software because it does not utilize a database to store metadata information. Instead, the software stores InDesign and InCopy files in a normal directory of the administrator's choosing, tracks article status and metadata using a flat XML file for each project ("publications" in Smart Connection parlance), and controls access to articles and layouts using the operating

[97] WoodWing Software, "New York Sun Daily Newspaper Chooses WoodWing," http://woodwing.com/en/Newsflashes/20080418_New_York_Sun (accessed June 4, 2008) and WoodWing Software, "Express Newspapers in the UK selects WoodWing," http://woodwing.com/en/Newsflashes/20071218_Express_Newspapers (accessed June 4, 2008).

system's built-in access control features. There are advantages and disadvantages to this approach that will be covered later in this quick review of the software.

Since there is no server software, installation is as simple as running the client install on each computer with InDesign or InCopy. The PDF manual provided with the software covers the install and configuration process in step-by-step detail. Overall, it should be relatively easy for even inexperienced users to set up Smart Connection Pro, with the only likely snag being the question of where to store the content files for the publication. Ideally this should be a directory that can be accessed by everyone on the local-area network; if no such directory or drive has already been set up, then someone with some basic Windows or Mac file sharing experience will need to configure a network location.

Smart Connection Pro: overall workflow structure

Smart Connection Pro is able to handle multiple publications; since most small magazine publishers only have one title to worry about, this review will assume a single-publication environment in Smart Connection Pro, though the process does not change very much for multiple publications.

Smart Connection Pro uses three concepts to organize articles and layouts. **Issues** are the top-level organizing structure (aside from Publications) and are used to group articles by issue or edition. **Sections** indicate where in the publication an article or layout should go; these would map to the various sections of the magazine, such as columns or the feature well. If a magazine wished, it could ignore the concept of sections entirely and simply put all articles and layouts into one unnamed section; it's up to the magazine how much detail it wants to put into creating Sections. Finally, **Baskets** represent the various stages of the editing and production process. There are two main uses for baskets: the first is to mark the status of an article, and the second is to restrict

access to the article based on where it is in the process. It is possible in Smart Connection Pro to restrict, say, the Copy basket to copyeditors only, but these restrictions are created and maintained through the operating system itself, using file-based permissions.

Smart Connection Pro: the editing and production process

Because both InDesign and InCopy can be used to start publications and drive the workflow process, Smart Connection Pro can be said to allow both a layout-driven workflow and a text-driven workflow. Since most magazines start with the text and only introduce layouts later in the editing process, this review will focus on Smart Connection Pro's text-driven workflow and interface elements related to a text-driven workflow.

The process begins in InCopy. First, the handling editor sets up InCopy for the current issue by selecting the correct issue or creating a new one using the Smart Connection palette in InCopy. As articles arrive, the handling editor opens the Word document in InCopy (which is able to import Word documents) and begins editing. As soon as the handling editor saves the InCopy document for the first time—which may or may not be after the editor is finished editing—the article will be filed in the Smart Connection system. While the handling editor has an article checked out, no one else can access the article via InCopy, but the art director can create a layout in InDesign and start adding articles without affecting the handling editor in InCopy. By default, designers cannot change the text of articles in the system unless they explicitly check out an article themselves. They can, however, alter layouts and access the last-saved version of an article for the purposes of copyfitting. The designer can also alert editors of layout changes so that editors can copyfit an article using InCopy's copyfitting tools.

While the art director is working on the layout, the text of an article can continue through the editing process. Once the handling editor has finished editing the piece, the editor checks the article into the system and sends the article to the next basket. At that

point, the next editor can start editing the text. As both editor and designer make and save changes to their respective documents, the other will be informed of the changes and asked if they want to merge those changes with their own document. Once an article has reached the final basket in the editing process, the art director knows that the piece has been finalized and can tweak the layout so that the article fits exactly.

Smart Connection Pro: proficiencies

The major advantage of systems like Smart Connection Pro is the tight integration with existing layout and editing tools, in this case InDesign and InCopy. By allowing designers and editors to work at their own pace while still remaining up to date on the other's progress, Smart Connection Pro makes it possible for art directors and designers to get involved in the workflow at a much earlier stage, and also perhaps to be more involved than the might otherwise be without the tight integration. Though none of the small magazines that participated in the study noted the integration of art and editorial as a major concern, systems like Smart Connection Pro offer new possibilities. For example, designers can make copyfitting suggestions to the editor at a much earlier stage, so that the editor can better determine what lines can be cut or what could be added. Another example where this integration would be helpful is the case of *Up Here*. Editor Aaron Spitzer noted that it was difficult to make proofreading changes at the same time the art director was tweaking layouts because there was only one computer running InDesign. A system like Smart Connection Pro would allow the *Up Here* editors to input those proofreading changes on any computer with InCopy, thus allowing the art director to stay at his computer.

More specifically to Smart Connection Pro, the software is very easy to use and install. The documentation that comes with the software is comprehensive yet easy to follow, and once users are acquainted with the basic concepts like Baskets and checking

out articles, using the software is quite simple. Anyone familiar with InDesign should have very little trouble understanding how the Smart Connection client for InDesign works, and the client for InCopy is even easier to use. Furthermore, both applications integrate with Smart Connection relatively seamlessly. For example, in order to save a new version of a file to the Smart Connection system, you simply save the file as you would a normal InDesign or InCopy document. The major learning curve will likely be with editors learning to use InCopy, as none of the magazines involved in the study had any experience with Adobe's copywriting application.

Smart Connection Pro: issues

Smart Connection Pro is quite clearly designed for fast-paced, deadline-sensitive environments where the parallelization of art and editorial workflows saves a lot of time and keeps staff from having to wait on other people in the process before they can do their own jobs. Small magazines would benefit from this parallelism to some extent, especially during crunch times near the end of publication, but not nearly as much at the beginning of the process. Magazines that publish fewer issues per year will get even less of an efficiency boost because by the time an article enters layout, most of the editing has already been done.

There is an opportunity, of course, for small magazines to simply start designing layouts earlier in the editing process to take advantage of the parallelization. However, most magazines in the study said that articles tended to enter layout either after copyediting or after the substantive editing. Because authors tend to work largely in Word, handling editors will have to send drafts back to authors in Word format, and as noted above, InCopy has no easy method of exporting back to Word documents. This is especially unfortunate given InCopy's relatively robust editing features, including the ability to track changes and leave inline notes that remain invisible in InDesign layouts.

Therefore, editors will have to make a choice: perform substantive edits in InCopy and get articles into the Smart Connection system early, but at the cost of having to manually export articles back to Word to send back to authors, and manually merge changes back into InCopy; or perform substantive edits in Word, eliminating the need to import and export drafts constantly, but keeping articles out of InDesign until much later in the process.

Finally, a minor issue with configuring user permissions: Smart Connection Pro handles access control via file system permissions in the operating system, a relatively high-level computing concept for most users. Because Smart Connection Pro does not offer an easy-to-use interface for user permissions in the software itself, the most likely scenario is that users will completely ignore the file system permissions, thus ignoring Smart Connection's access control abilities. But this means any InCopy user can edit articles at any stage in the editing process. In practice, it is unlikely the staff of a small magazine will be inclined to sabotage the production cycle by attempting to edit an article in the wrong basket. That still means that mistakes can be made because editors are not restricted to their specific baskets—for example, a copyeditor mistakenly making copy edits to an article before the handling editor has made their substantive edits.

Differences between Smart Connection Pro and more advanced packages

Some of Smart Connection Pro's issues are addressed by the more expensive software solutions like QPS and WoodWing's own Smart Connection Enterprise. Both QPS and Smart Connection Enterprise allow the use of other programs, including Word, for editing text.[98] As a result, the issues related to using InCopy for all editing purposes are mostly solved, as editors can choose to continue using Word while still using the

[98] Erik Vlietinck, "IT Enquirer Report on Editorial Workflow Systems," *IT Enquirer*, 2007. Available via http://www.it-enquirer.com/main/ite/more/qps_sce_k4_report/ (accessed February 6, 2008).

workflow system. Softcare K4 does not allow the use of Word as an editing application in the system, but the K4 system does recognize Word documents and will import them properly. Again, it is unclear whether it would be easy or even possible to merge changes from authors in K4. All three systems also solve the problem of handling user permissions via the operating system's file permissions structure. QPS, K4 and Smart Connection Enterprise all have more advanced access control features for determining which users get access to what parts of the workflow.[99]

The major downside of more advanced solutions, however, is that they require additional IT resources, such as a file and database server, and more extensive installation processes to run properly. More advanced systems will also cost more than simpler software like Smart Connection Pro, as explained earlier.

Bricolage: a web-based content management system

Bricolage bills itself as an "enterprise-class content management system," but it differs from the enterprise-class integrated solutions featured earlier in this report in a number of fundamental ways. First, Bricolage is designed with web publishing in mind, and is able to publish content in HTML or XML format for use on the web. Second, Bricolage is freely available for download and is open source. The only impact on a magazine's budget will be the time and money spent on installing and maintaining a Bricolage install, and should the magazine need to make modifications to how Bricolage works, it is theoretically possible to either use third-party code made available by others or else make the modifications necessary to the Bricolage source code. Again, the only

[99] Quark, "Quark Publishing System: How it works," www.quark.com/products/enterprise/modules/qps/pdf/QPS7_HowItWorks_US_Web.pdf (accessed July 18, 2008), 1; Softcare, "Softcare K4," http://www.softcare.de/publishing-solutions/softcare-k4/index.html (accessed July 18, 2008); WoodWing Software, "Smart Connection Enterprise Brochure," http://www.woodwing.com/userfiles/file/Brochures/Smart%20Connection%20Enterprise/WoodWing_Smart_Connection_Enterprise_brochure_EN.pdf (accessed July 18, 2008), 2.

cost is the time and money required to make those modifications; there are no problems regarding new license requirements or going back to the vendor to purchase additional plug-in functionality. These are potentially important considerations for any magazine with an unorthodox workflow system that does not completely fit within Bricolage's model. They are also interesting from the perspective of a possible development effort towards providing a workflow solution aimed at small magazine users; if Bricolage's codebase offers many of the features small magazines need, then modifying the code to further tailor the software to small magazines may be a viable option.

Bricolage bears some similarity to web frameworks and applications like Zope/Plone in that it is intended primarily as a tool for web-based content management and publishing. One key difference is that Bricolage does not have its own web server and requires that the task of actually serving a website be left to another software package. For the purposes of print publication, the lack of a web server is not a major problem, though it is worth noting for magazines that wish to publish editorial content to multiple media. Another key difference is Bricolage's robust workflow engine, designed not simply for storing information like most web-centric CMSes, but also to model the editing process. Major users of Bricolage include the website for MacWorld magazine and Salon.com.[100]

[100] David Wheeler, "Content Management with Bricolage," *O'Reilly perl.com*, August 27, 2004, http://www.perl.com/pub/a/2004/08/27/bricolage.html (accessed June 4, 2008).

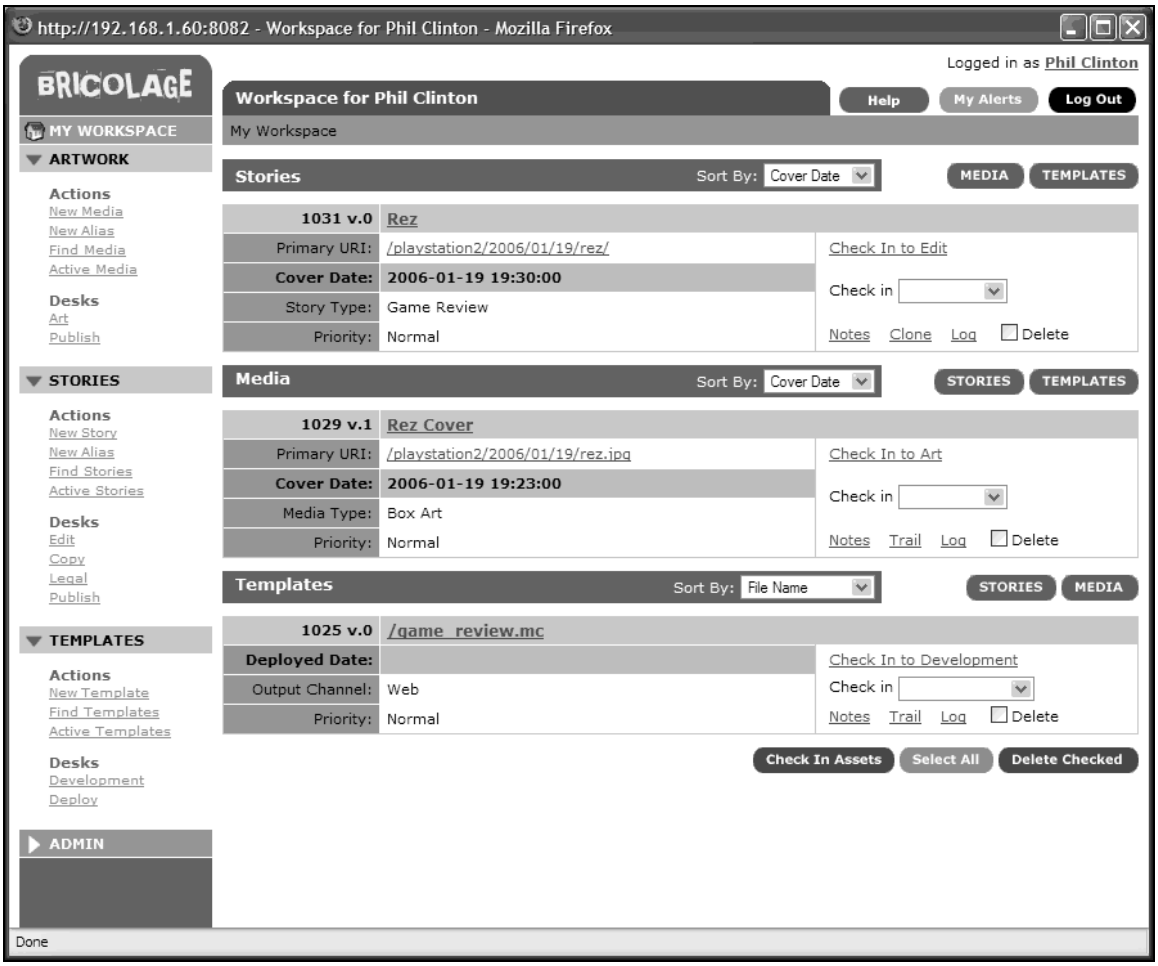

Figure 2. A user's home workspace view in Bricolage, showing all the assets the user has checked out. Licensed under a Creative Commons Attribution-Noncommercial 2.0 Generic license (http://creativecommons.org/licenses/by-nc/2.0/) from the Bricolage website. (http://www.bricolage.cc/docs/screenshots/)

Bricolage: installation

Though the Bricolage software is free, it does require a level of technical expertise to install. This caveat alone can take Bricolage out of consideration for many small magazines, since most lack a dedicated IT staff. *Briarpatch* rejected Bricolage because editor Dave Mitchell felt it would be too difficult to get the software up and running.[101] Even evaluating Bricolage can be difficult, though the recent creation of a VMWare virtual machine image makes the process of installing an instance of Bricolage for testing

[101] Mitchell, e-mailed questionnaire.

purposes much simpler. Even the VMWare image is not completely trouble-free, however; the image available as of March 2008 requires some troubleshooting to get it working.

Actually setting up Bricolage for normal use is even more involved and requires proficiency with Linux, building applications from source code, and understanding the nature of dependencies as well as installing or compiling them. This is clearly beyond the capabilities of most small magazines, and any small magazine looking to adopt Bricolage will almost certainly require outside assistance to get the software running.

The picture changes if a magazine has a technically-minded staffer with knowledge of Linux. An installation procedure that would be difficult for those with no technical background is accessible to those with an understanding of how Linux works. Because of the presence of easy-to-install packages for operating systems branched from FreeBSD or Debian, related distributions aimed at users with relatively little Linux experience, such as Ubuntu, are ideal for magazines looking for the easiest (though still somewhat difficult) way to get started with Bricolage.

Bricolage: overall workflow structure

Bricolage can handle both art and editorial workflows. Since the small magazines in the study often had a single person handling all art duties (and in two cases had an editor who also handled art direction and layout), this analysis will focus on the editorial workflow tools. Bricolage's art workflow works in roughly the same manner, however.

The primary objects in the editorial workflow are **Story** elements, which contain the text and accompanying metadata of an article. Bricolage comes with a standard Story model that includes a number of metadata fields for information like publication dates, but it is possible to create new models that extend the Story model if a magazine needs

more metadata fields. Bricolage comes with several Story sub-types of its own, though it is more likely that a magazine will define its own sub-types based on the standard editorial mix for an issue—for example, a magazine may have a Column story sub-type for individual columns, or a Review sub-type for capsule reviews in a Reviews section.

Story elements contain the content in the Bricolage editorial workflow. A **Workflow** applies to specific element types and is defined as a series of desks. **Desks** define the various stops in the editorial workflow where work needs to be done. To illustrate the idea behind desks, Bricolage comes with a default four-desk setup that will work for most magazines: an Edit desk, a Copy desk, a Legal desk, and a Publish desk. As with content elements, it is easy to modify Bricolage's workflow by adding or removing desks—for example, many small magazines do not have an explicit legal approval stage, so they would probably remove the Legal desk from the workflow.

Finally, once an element has made it to the final desk, usually called the Publish desk, it can be published to an **Output Channel**. Output Channels can be thought of as a way of transforming content in the Bricolage system into files that can then be used outside of Bricolage. Because Bricolage was designed primarily for web publishing, the default Output Channel is the Web channel. When a Story is published to the Web channel, the content of the Story element is transformed into HTML that can then be served on a web server. This transformation is done via an HTML generating engine called Mason, but other template engines and scripting languages are available for use. Bricolage also offers the ability to publish to XML. Bricolage does not have the ability to export to file formats commonly used in print production, such as Word, QuarkXPress or InDesign documents, though it is theoretically trivial to output articles as plaintext files that can be imported into those programs.

Bricolage: the editing and production process

Bricolage does not have a built-in concept of separate issues, though a magazine can assign categories to Story elements that map to issues. When a magazine receives the first draft of an article, the handling editor starts a new story by using the New Story action. Once a new Story element is created, Bricolage automatically moves it to the first desk in the system—the Edit desk, in the default workflow—and checks it out for the editor's use so that no one else can access or edit it. At this point, the handling editor can add the text of the article to the Story element.

Bricolage defines the content of a story as a document tree containing several sub-elements; the nature of this document tree is defined when you set up a Story sub-type for the first time. For example, a magazine may wish for feature articles to have a deck, a body and a short author bio, while capsule reviews would not have a deck but would have a five-star rating field. This structured approach to creating and managing article content is far more complex than what the small magazines in the study are used to, and it is unclear whether adopting the Bricolage approach would be a productive gain or a hindrance.

Once an editor has finished editing an article on the Edit desk, the editor checks the story back into Bricolage and moves it to the next desk in the process, whereupon a different editor will perform whatever tasks they need to perform. Story objects move from desk to desk until it reaches a desk with publishing capabilities—in the default workflow, the Publish desk—and the editor at that desk approves it for publishing, at which point the Story element is converted into a flat file for use elsewhere. Theoretically, the editor can then pass on the flat file to the art director to insert into a QuarkXPress or InDesign layout.

Bricolage: proficiencies

Bricolage is highly customizable and fairly flexible, meaning it can be modified to work with a wide variety of workflows. For example, *Spacing*'s workflow is complex compared to the other small magazines in the study, in that any given article will have two substantive editors, and any given editor could be working on as much as a third of an issue as either the primary or the secondary handling editor. *Spacing* could model this workflow in Bricolage by creating Story sub-types that added metadata fields for primary and secondary editor assignments, and then creating a workflow with a desk for each individual editor as well as a copy desk and a proofing/publish desk. An additional benefit is that Bricolage makes it relatively easy for each user to see what tasks are left to be done. The default view for each desk is a list of all the stories that have yet to move on to the next stage of editing, and each user's workspace shows all the assets the user still has checked out for editing.

Finally, for magazines that wish to publish to both print and the web, Bricolage is already set up to handle web publication. Publishing to the web is a relatively straightforward process that involves very little work compared to creating the pages manually or taking content out of a layout file or PDF and inserting it into another content management system manually. In fact, it is possible to publish an article to multiple Output Channels at the same time, meaning publishing to the web requires no additional work or thought at all from issue to issue.

Bricolage: issues

There are a number of quirks in Bricolage's workflow that make sense in its original corporate, web-focused context, but can be problematic when asked to manage an editorial workflow for a print-focused small magazine. Overall, Bricolage is simply too

difficult to set up and use for most small magazines. Aside from the difficulty of the initial install, there are three main problems regarding ease of use.

The first is the difficulty of editing the text of an article. Bricolage's structured document modelling works well for web articles, where certain features like pagination and the placement of inline elements, like images and pull quotes, needs to be explicitly defined in the text. However, in print, most of those issues are handled by the layout program, and so do not need to be explicitly defined in the text. Furthermore, Bricolage's default settings create an overly oppressive document structure that requires paragraphs in the text to be defined as individual objects, making it difficult to edit them in the Bricolage interface. Editing an article is really more like editing a series of paragraphs in separate text boxes if you use the normal view. If you use the Bulk Edit view, you can edit the whole article at once, but the view also reveals Bricolage's internal structural markup language.

This is somewhat acceptable in a web-publishing environment, where users are expected to have a basic understanding of structured markup languages like HTML. In print, this is not the case. Furthermore, if an editor damages the structural markup in the process of editing the text, it can cause entire paragraphs to disappear without warning; clearly this sort of data loss due to user error is unacceptable. It should be noted that the general concept of having a structured document tree is not inherently faulty; advanced layout programs like Adobe Framemaker utilize similar concepts, and every website is built upon a body of structured HTML documents. Bricolage's interface for accessing and editing that document structure, however, turns what could be a plus into a minus.

As an editing platform in general, Bricolage leaves much to be desired. There is no facility at all for making editor's marks or leaving notes for authors, making the process of editing initial drafts in Bricolage much more difficult than in Word.

Furthermore, if authors don't have access to the Bricolage system (a likely situation), the editor has to pull text out of Bricolage when sending an article draft back to an author, and then has to reintegrate the new draft from the author's Word document. It is unlikely a magazine would choose to allow authors into its Bricolage system, if only because it would be far too tedious to create system users for each writer and assign the correct permissions.

The second problem is the interface for setting user permissions. Ideally, Bricolage would allow only specific people to access desks in the workflow—for example, a magazine may not want copyeditors to see articles in the Edit desk. Theoretically, Bricolage allows for this sort of permissions-based access to various desks in a workflow. In practice, however, the user permissions model is overly complicated and the permissions interface non-intuitive. The average small magazine editor will likely spend a great deal of time and effort figuring out the simple task of restricting desks to specific editors, assuming they discover the functionality at all. The documentation given for Bricolage's permissions system does not explain this task in step-by-step detail, and it took me several hours of reading the documentation to figure out how to assign desk permissions to a user group. I am a relatively computer-savvy person, having built web sites using PHP and Python; the average small magazine editor will likely have even more trouble than I did.

Having such a deep, complex permissions model is great if the user truly needs such fine-grained permissions—corporations with hundreds of employees are examples—but for a small magazine that's unfamiliar with workflow software, such a complex system can be daunting. In the end, Bricolage is perfectly usable without defining any user permissions at all, provided a magazine does not mind using the honour system to restrict people to specific desks. Like integrated workflow solutions

like Smart Connection Pro, however, even if a magazine trusts all its users not to make changes outside their jurisdiction intentionally, there's always the chance for mistakes.

The final problem is integrating Bricolage with a print production system. This problem effectively prevents most small magazines from using Bricolage in a meaningful capacity for print publication. As stated earlier, Bricolage publishes finalized Story elements to flat files. The original intent was that Bricolage would output finalized HTML to the root web directory on a web server, such that publishing new articles to the website would be a one-click operation. However, for print production, there is no one-click export to layout software like QuarkXPress and InDesign. Though it is theoretically possible to construct an XML workflow that could handle such a task, currently there is no readymade solution available.

Many of these problems could be mitigated or fixed with outside assistance—some sort of Bricolage integration service aimed at small magazines, for example—but other problems, like the relatively poor editing feature set, would require changes to the Bricolage software itself.

Open Journal Systems (OJS): a web-based academic publishing workflow system

Open Journal Systems (*OJS*) is a workflow management package designed for academic journals. Created as a low-cost, easy-to-use solution for journals pursuing an open access philosophy, OJS is able to manage both the workflow and the content of an academic journal. One of OJS's major features is the ability to model the sorts of comprehensive peer review processes most journals use to filter submissions fit for publication. OJS also offers the ability to publish articles and issues to the web.

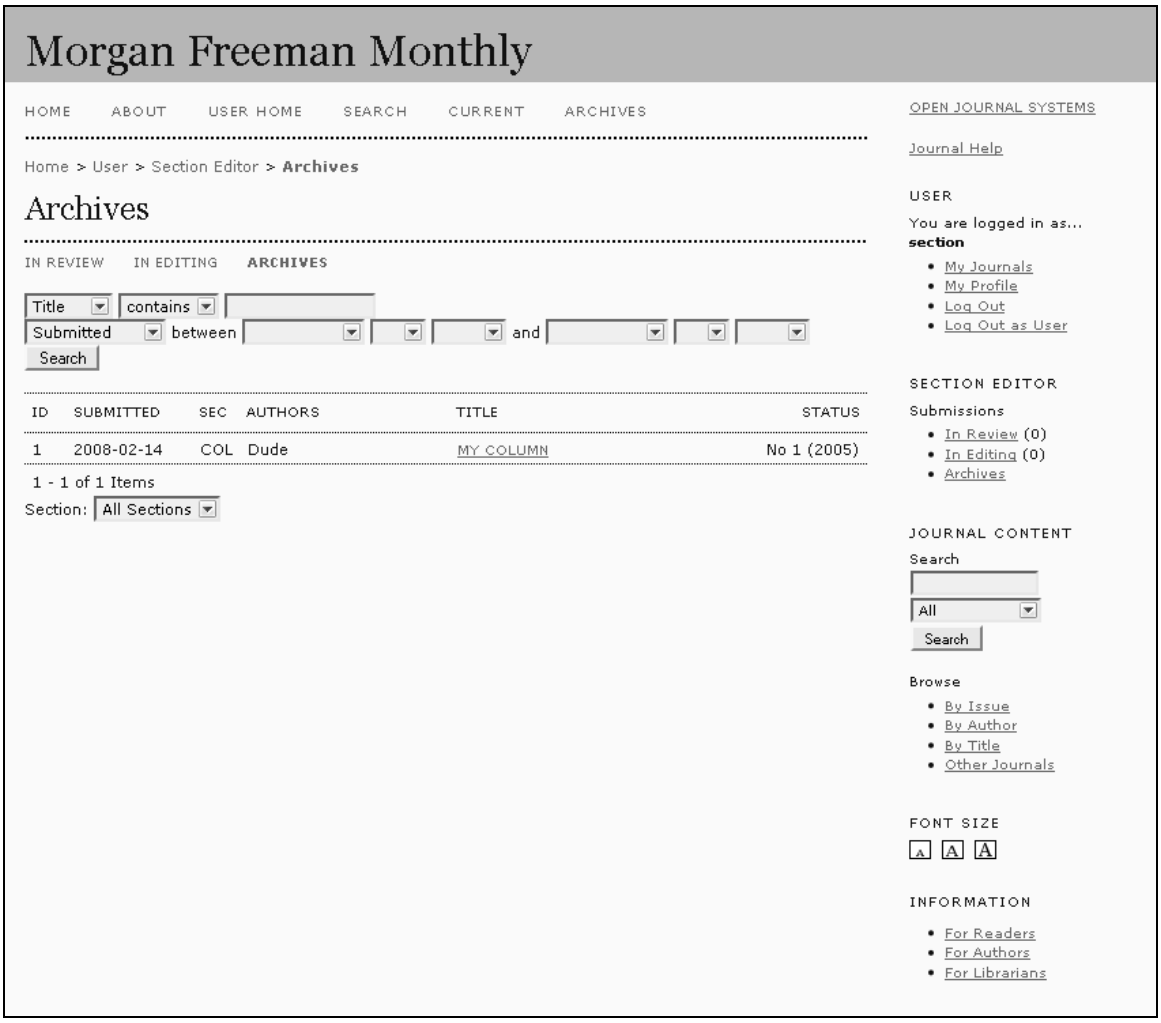

Figure 3. The section editor view in Open Journal Systems. © Public Knowledge Project, with permission. (author's own screenshot)

OJS is open source and offered free of charge by the Public Knowledge Project, a joint venture between the University of British Columbia, Simon Fraser University, and Stanford University. Built using widely used technologies like PHP and MySQL, OJS should also be relatively easy for programmers to modify and extend to suit a publication's particular needs. OJS installations live on web servers and all interactions are handled through a web interface, meaning there is no need for users to work in the same office or use the same local-area network.

Open Journal Systems: installation

Because OJS is based on web technologies like PHP and MySQL, installing it requires some familiarity with the basics of managing a web server. However, aside from this requirement, OJS is relatively easy to install—slightly more difficult than products like Smart Connection Pro, but much easier than Bricolage. One widely used web application with a similar level of complexity to its installation procedure is Wordpress, a popular blogging application. The open source project boasts that most users can install the program in five minutes,[102] and the basic steps are very similar to OJS: create a database, fill in the database details in the configuration files, upload the whole package to your webspace, and run the install script in your web browser.

OJS offers installation instructions and additional setup considerations in its README file. Assuming the web server has the necessary components—nearly every web hosting service offers PHP and a database implementation, usually MySQL, so this is unlikely to be a problem except in the most marginal of hosting environments—all that is needed is an FTP client that can handle file permissions. Setting up a journal in the software actually takes longer than the installation itself, but the provided documentation contains a step-by-step guide and the setup process itself is fairly easy to follow.

Open Journal Systems: overall workflow structure

Of all the workflow software packages reviewed, OJS has by far the most detailed and complex structure. Many of OJS's features, like handling subscriptions or providing rich reading tools designed to link to other journals or academic repositories, are irrelevant for the purposes of magazine publishing. The same holds true for many of the

[102] Wordpress, "Installing Wordpress," http://codex.wordpress.org/Installing_WordPress#Famous_5-Minute_Install (accessed June 4, 2008).

workflow structures OJS imposes; in many cases the structures are unnecessary for most magazines (such as the comprehensive peer review system) or else don't fit the needs of magazine publishing very well (such as the heavy involvement of authors at many points in the workflow). It is possible to route around certain aspects of the workflow, however. For example, section editors can ignore the peer review features of OJS if they wish, and limit themselves to the editor tools alone.

OJS allows for multiple journals per install; this review will only look at single-title operation. Each journal is divided into **Issues**, which can be organized in a number of different ways, but notably not by month/season (only year, volume, number and/or title—again, an artifact of OJS's origins in academic publishing). Each issue is divided into **Sections**, which map to the various sections of the magazine. They are used mainly as containers for the different types of content, as well as an organizing structure for issues on the website OJS produces.

OJS's workflow is fixed; unlike the other workflow systems reviewed thus far, a publication cannot tailor the workflow to fit their own process without modifying the OJS source code. Instead of defining each step in the workflow as if it were a linear process, OJS splits tasks up by **Role**. There are seven roles in OJS with substantial editorial duties; multiple users can share the same role (for example, there can be several Section Editors) and users can have multiple roles, though the interface for adding roles to a user is somewhat non-intuitive. The seven roles are as follows:

- **Authors** submit an article for publication. They also receive updates on the submission during the peer review and editing process.
- **Editors** handle administrative duties for the journal and direct submissions to the proper Section Editor. Editors can also act as Section Editors during peer review and editing.
- **Section Editors** handle the task of assigning peer reviewers to an article and editing the article as necessary. They also approve or deny submissions after the

peer review process is complete. Finally, section editors assign copyeditors, layout editors and proofreaders to an article (or act as one or all of those roles) and monitor the editing process.

- **Reviewers** handle the peer review aspect of the publication. They read articles and give feedback as to what needs to change and whether the article is suitable for publishing, needs revisions, or should be rejected.
- **Copyeditors** handle copyediting duties. Authors also copyedit their own work.
- **Layout Editors** take the article and create a set of layout files. OJS expects HTML and/or PDF versions by default, and will offer both through the journal's website (as a readable page or as a download respectively).
- **Proofreaders** proofread the layout files. Authors also proofread their own work.

Open Journal Systems: the editing and production process

Because the workflow of a magazine is different from that of an academic journal, this review will not explore in detail any features relevant solely to academic publishing.

The author starts the process for a given article by submitting the article through the OJS system. OJS is unique amongst workflow systems in that authors are explicitly included in the process. There are advantages and disadvantages to including authors in the OJS system. As with Bricolage, adding authors into the system would be tedious for magazine editors, considering the potential number of contributors per issue; however, it is possible for authors to register with OJS themselves. Whether or not having authors manage their own user accounts is too much of a burden is debatable; for freelancers writing for several publications, maintaining accounts with several magazines may become an annoyance. Whether a magazine even wants authors to be present in their internal workflow system is also uncertain, but OJS requires that authors be a part of the system. A workaround to this issue would be to give every editor in the system the Author role—not ideal, but at least usable.

The author submits an article by filling out the relevant metadata and attaching a file to the submission containing the text of the article. OJS does not deal with the text of an article directly; instead, everything lives in external files that everyone in the system accesses in order to view or edit documents. OJS keeps track of an article's state throughout the workflow by creating and storing versions of the original file. By using a file-based system rather than storing the text directly, OJS sidesteps many of the concerns regarding web-based editing in software like Bricolage and Basecamp.

Once the author has filed a submission, the editor assigns the article to a section editor (or themselves) for review. The section editor then shepherds the article through the review and editing stages. Academic journal section editors would begin assigning peer reviewers at this point; since magazines generally do not have peer reviewers, the section editor would limit their interactions in the Review pane to the Editor Decision section, labelling the article as in need of further edits or completed as needed. The section editor can send the author the revised document with marked edits through OJS, and the author can similarly upload new revisions to the OJS server and notify the editor.

To signal the end of substantive editing, the section editor then marks the submission as accepted and sends the relevant file version to copyediting. The section editor oversees the copyediting, layout and proofreading stages of the process from the Editing pane. The section editor assigns a copyeditor to the article and sends it to the copyeditor using the Request feature. After copyediting, the section editor assigns a layout editor and requests a galley; after layout, the section editor assigns a proofreader for a final read. After that, the article is ready for publication.

Open Journal Systems: proficiencies

OJS has several features that seem tailored to collaborators working virtually from multiple locations, a situation shared by many small magazines. For example, OJS's built-in e-mail alert system is an efficient way of managing communications between users and keeping users up to date on what they need to do next. The default e-mail templates are also good at describing how to perform the next step in the process, which is helpful for new users.

OJS's use of files to store the contents of an article, versus saving the plaintext itself, has several advantages. By accepting files instead of trying to manage content directly, OJS allows users to choose which editing and layout programs they wish to utilize, meaning no magazine has to lock itself into a certain software package. Moreover, the problems Bricolage and Smart Connection Pro had with standard tools like Word (in both cases) and InDesign (in Bricolage's case) do not exist in OJS. Editors and authors can feel free to continue editing in Word without worrying about import or export problems, and designers can import text into InDesign or Quark as before, without worrying about problems with text formatting. There are problems with this approach to content management, however, which will be covered in the next section.

Finally, OJS's workflow setup incorporates the entire publishing process from start to finish, including proofreading and layout. Many packages fail to model the substantive editing stage very well, especially the correspondence between editor and author, and Bricolage essentially stops at the print layout stage, having assumed that whatever files it produces represent the final product when in fact they do not. By contrast, OJS includes all the major roles in the editorial cycle and handles communication and workflow traffic between all participants. Communication between substantive editing, line editing, layout and proofreading stages are all covered.

Open Journal Systems: issues

The major problem with using OJS in its current form in a small magazine environment is that OJS is clearly designed for academic publishing. The differences between the two worlds are significant enough that for a small magazine to use OJS, it either has to accept many of the quirks of academic publishing using OJS, work around or adapt to them, or modify the OJS code to suit its needs. The latter, luckily, is an available option since OJS is open source; this option would not be available with software packages operating under a closed-source license. The situational differences would not be so much of a problem if OJS was more flexible, but all the workflow components in OJS are hard-coded, and thus are difficult to modify without soliciting a programmer. In practice, a publication would likely work around quirks in the software more often than not.

A major example of OJS's structural rigidity is the workflow model OJS uses. Most workflow solutions allow the publication to customize the workflow to suit their needs—a publication can define all the stages of the workflow as necessary, and assign users to each stage to restrict who can do what. OJS, by contrast, allows very little customization. For example, because OJS splits user permissions into roles, it may be necessary to give certain users in the system multiple roles to gain access to certain parts of the workflow. The interface to do this, however, is not immediately obvious.[103] Another example is that OJS asks authors to submit articles to the system themselves. This either requires that editors be assigned the role of author, which is not very intuitive

[103] In order to assign an additional role to a user, anyone with access to the Journal Management options must click "Enroll a User from this Site in this Journal," select the additional role under "Enroll user as," and re-enroll the relevant user. This process is repeated for every additional role to be assigned.

but at least a simple workaround, or that authors be allowed into the system—an option that may be undesirable in terms of user management for several reasons.[104]

OJS also enforces a specific workflow with pre-defined stages, and while it does allow deviations from its standard academic-journal workflow, this largely has to be managed through the use of workarounds. For example, *Spacing*'s use of a primary and secondary substantive editor would work in OJS, but requires creative use of the peer reviewing system to manage this interaction (where the primary editor acts as a Section Editor and the secondary editor as a Peer Reviewer). Similarly, many magazines will have multiple proofreaders and/or proofreading stages, but OJS requires exactly two proofreaders (the author and a designated proofreader); ideally OJS would allow magazines to assign any number of proofreaders. Layouts are article-specific, meaning layouts with multiple articles do not quite fit into OJS's workflow model; storing and managing multi-article layouts would probably be done outside OJS. None of these issues would necessarily prevent a magazine from adopting OJS as is, nor would it necessarily be difficult in theory to modify OJS to remove some or all of these issues (though without a comprehensive code review it's difficult to make any solid claims for or against the ease of modification; such a code review is outside the scope of this report). However, modifying OJS would be far preferable to asking magazines to adopt various workarounds to operate OJS efficiently.

[104] OJS offers two options for creating users: the Journal Manager can create all users and assign all roles, or users can register themselves as Readers, Authors or Reviewers. Choosing the former option means Journal Managers have to handle the tedious task of giving all authors a user account, which can be laborious for magazines with a substantial number of contributors. Choosing the latter option means that anyone can register as an author, including people who aren't supposed to be authors in the system. The internet is a sufficiently compromised venue that any public form, such as a user registration form, should be assumed to be the target of spammers, hackers, and other unsavoury users. The obvious workaround is to put the magazine's workflow system behind a password-protected gateway, and give the password only to authors; this requires a small amount of website management knowledge, but would likely be the best scenario for magazines that wished to include authors into the system.

Other eccentricities have less severe effects, but working around them still requires some effort. This is especially obvious when editing the settings for a Journal—many of the configuration options available, such as academia-focused text fields for journal sponsors and peer review guidelines, the various academic indexing and reviewer database options, and metadata harvesting, are irrelevant to magazine publishing. Of course, publications tend to go through the configuration process only once; other quirks pop up far more often. Author submissions require metadata that may be unnecessary (like abstracts) or would be better handled elsewhere (like article titles or sections). Another eccentricity is the required step of an editor assigning submissions to section editors. It is a minor step but one that probably would be eliminated in a magazine context, since generally authors correspond directly with section editors from the start.

Finally, another word must be said about the use of files to store content versus storing the article text itself. As previously stated, one advantage to using files to store content is that the magazine can utilize their choice of editing and layout tools without worrying about whether OJS will work with them. However, there is a downside as well: it also means that OJS itself does not work as an editing or layout tool because it never touches the text. As a result, certain features are hard to implement and simply don't exist in OJS, like the ability to see what's changed between drafts of an article—instead, OJS relies on the editing facilities of whatever program was used to generate the files OJS stores. From a more general perspective, it also means that a lot of work necessarily lies outside OJS's purview—the layout stage, for example, is simply a checkbox in OJS that gets marked off when the layout editor has taken the article text out of the attached file in OJS, laid it out in a layout program, and posted the final PDF and/or HTML file. OJS's presence during the transformation is largely a bookkeeping and organizing one, which is valuable on its own, but may not be as useful as workflow systems where the content is stored in a form accessible to the system as editable text.

Overall, the current iteration of OJS seems to be well suited to the task of managing an academic journal's workflow, but is not ideal for the task of managing a small magazine workflow. Magazines can utilize OJS in its current state, but will likely have to accept a number of workarounds to do so. Furthermore, it is possible with modification to adapt OJS to more closely meet the needs of small magazines. Ideally, OJS should be modified to be more flexible in allowing individual magazines to define the parameters of the workflow, such as defining workflow stages and permissions for each stage, as opposed to simply changing the particulars of a hard-coded workflow. The roles model, used to define user permissions in the system, may also be too rigid to handle varied magazine environments where users may have differing responsibilities that do not map neatly to pre-defined roles. Functionality similar to the way Bricolage handles flexible workflows, in which each stage of a workflow is modelled as a desk to which specific users have access, would go a long way towards making OJS easier to adopt by a wide array of magazines. Streamlining some of OJS's functionality would also help, such as removing unnecessary complications like the author submission checklist, and much of the academic journal language.

Recent developments: the OMMM project

In a bid to adapt OJS to the needs of small magazines, the OMMM project, hosted by the Canadian Centre for Studies in Publishing at Simon Fraser University, has laid out some guidelines for extending OJS to better fit a small magazine environment. Some of these changes include removing language and features specific to the academic publishing environment.[105]

[105] John W. Maxwell, "Extending OJS into small magazines: The OMMM Project", *First Monday*, 12 no. 10 (1 October 2007), http://www.uic.edu/htbin/cgiwrap/bin/ojs/index.php/fm/article/view/1962/1839 (accessed July 17, 2008).

Basecamp: a web-based project management application

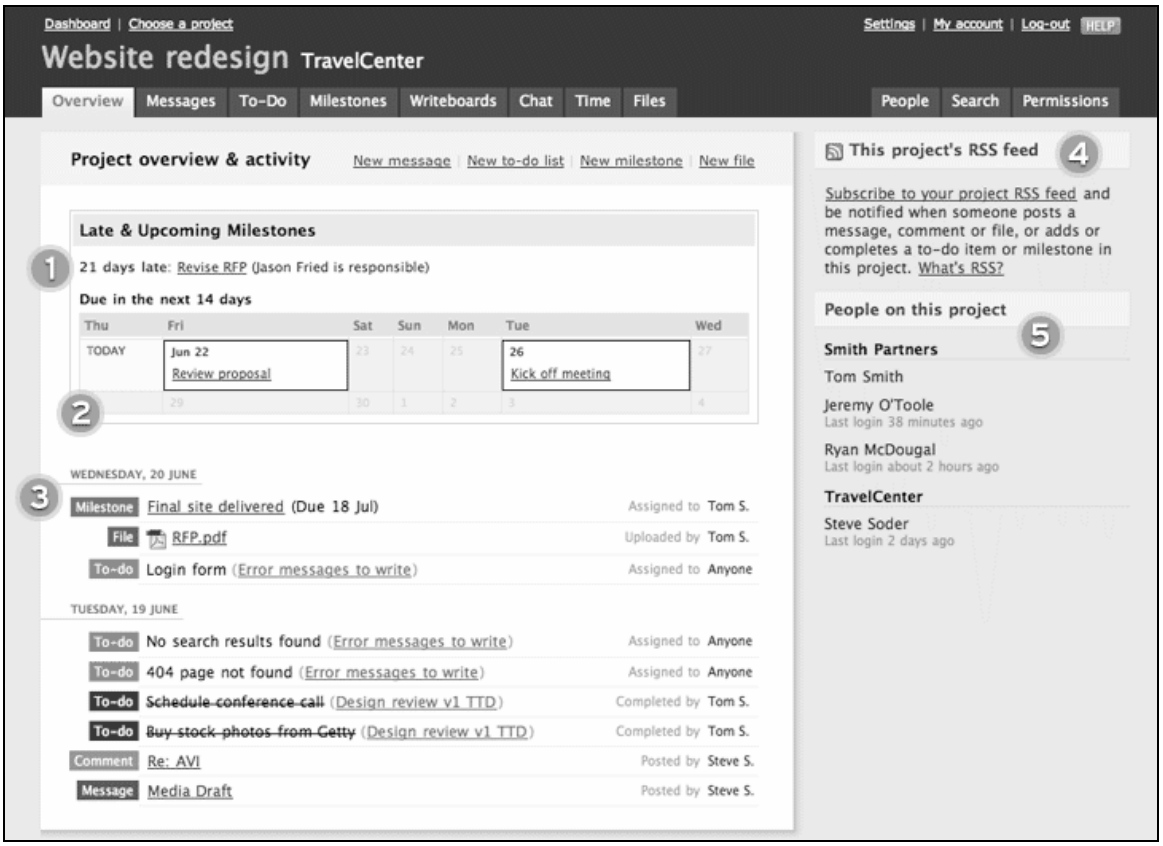

Figure 4. Project overview page in Basecamp, listing various milestones and to-do items in calendar form, alongside files and communication items posted. © 37signals, with permission. (http://www.basecamphq.com/tour)

Created by web design firm 37signals in 2003, *Basecamp* was originally designed as a project management tool for the company's internal use. Based on the response 37signals received from its design clients regarding the software, the company decided to release the application to the general public.[106] Since then, a number of companies of varying sizes spanning many fields have adopted Basecamp, including Digital Web

[106] Farhad Manjoo, "The next Web revolution," *Salon*, August 10, 2005, http://dir.salon.com/story/tech/feature/2005/08/10/37signals/index1.html (accessed June 4, 2008).

Magazine, an online magazine, and MacZealots, a community-driven website for Apple users.[107]

Because 37signals originally developed Basecamp for use in a web development environment, and not for the purposes of magazine publishing, the application lacks several features that other workflow software solutions have, such as integration with layout software. However, Basecamp appears to be flexible enough to handle many types of projects, including magazine production; *Briarpatch* is already using Basecamp to manage their editorial workflow.

Basecamp: installation

Basecamp is hosted by 37signals and is accessible via any web browser, so there is no installation process. A publication simply signs up for an account; once the account has been activated, a Basecamp site is generated automatically for immediate use. Setting up Basecamp for use is as simple as creating user accounts for staff and starting a Basecamp project.

Basecamp: overall workflow structure

One reason why Basecamp is useful for such a wide variety of situations is because the application is extremely flexible. Basecamp is essentially a collection of tools that together offer several ways of managing projects. **Message Boards** allow staff to communicate with one another; **Milestones** are where staff members keep track of deadlines past and upcoming; **To-Do Lists** help staff keep track of what tasks need to be done for parts of a project; **Chat** allows for real-time text conversation and collaboration between staff members. Basecamp also offers **Writeboards**, a document repository and tracking tool, and a general file storage repository.

[107] 37signals, "Basecamp example uses and case studies," http://www.basecamphq.com/examples (accessed June 4, 2008).

Basecamp does not attempt to enforce any form of workflow, and indeed should be considered more of a planning and collaboration tool. For example, Basecamp's user permissions are fairly simple; people within a company can be granted access to projects, and within those projects they can add and edit everything. This means certain access control tasks, like restricting editing privileges on a first draft to the handling editor, are impossible in Basecamp. There is also no concept of workflow stages, nor is it possible to enforce one in the software; determining who in the system is allowed to edit a document at what point in time is essentially a matter of adhering to an agreed-upon convention, rather than enforced by the software.

As a result, publications essentially devise their own workflow, and agree upon a set of common practices to manage the flow of documents from editor to editor and the tracking of the editorial process. The Basecamp tools merely facilitate the expression of those common practices. For example, if a publication decided that adherence to and notification of upcoming deadlines was an important issue, that publication could define a large number of milestones that mapped to every stage of the workflow process, for each article in an issue. A less structured approach would be simply to define Milestones for every stage in the workflow, but for all articles at once—a publication could simply create one milestone for when all articles should be in copyediting. Alternatively, a publication may simply choose not to enforce milestones at all, instead tracking the progress of individual articles using to-do lists.

Basecamp: proficiencies

Basecamp's greatest assets are its simplicity and ease of use. Installation is as simple as signing up for a Basecamp account, and is by far the easiest of all the software packages reviewed. The Basecamp tools are straightforward and have very simple, intuitive interfaces. Novice users have access to a number of comprehensive

demonstration videos that explain the features of each tool, but chances are most internet-literate users will have no problems using Basecamp.

Another advantage to Basecamp is it's fairly easy for a publication to adopt without having to worry about hiring additional IT staff or buy extra software or computers. It is theoretically possible to use Basecamp for free, though most publications will likely opt for at least the Basic plan, which costs $12.95 a month and includes 250MB of built-in file storage. Even with the monthly fee, there are no additional expenses. There is no server infrastructure to buy, no web hosting to pay for, no software to install, and no additional staff to maintain any computer infrastructure.

Finally, Basecamp was designed in such a manner that companies can adapt the software to their way of working, instead of having to adapt to how Basecamp works. The tools come with no prior assumptions about what they will be used for. For example, a publication could use to-do lists for any number of functions, such as agendas for editorial meetings, editing checklists for individual articles, long-term planning checklists, lists of stock photography to acquire, or a combination of any or all of the above. Publications can also decide how much of their workflow will be handled in Basecamp—if a publication decides only to use Basecamp for its message board to send notes to staff members, that's just as feasible as another publication fully embracing all of Basecamp's features and using the application to assist with every part of the editorial workflow.

Basecamp: issues

In sharp contrast to OJS, Basecamp does not attempt at all to enforce a workflow. For example, enforcing any sort of access control—restricting which individual users can edit an article—is not merely difficult, it is impossible. Basecamp does not have a built-in method for signalling who the next editor is in the editing process, or explicitly passing a

document on to the next editor. *Briarpatch* handles this sort of traffic control by keeping track of article assignments in a writeboard, and simply having people contact each other when it is time to move an article to the next stage of editing.

In many ways, Basecamp encourages exactly the sort of informal, ad hoc collaboration and communication that other workflow software packages try to prevent, or at least keep to a minimum. Indeed, a magazine looking to add structure to their current workflow may find that Basecamp does not do enough. If anyone can edit writeboards, then enforcing access to writeboards must be done informally, outside of Basecamp. If anyone can declare a milestone finished, then everyone must agree that a milestone has been met, or else one person must be assigned the informal role of schedule keeper, and everyone else must agree not to change the milestones.

The problem with such an arrangement is it is not much different from the current situation of having one person—usually the editor-in-chief—act as a hub for all editorial interactions. The advantage to using Basecamp over manual tracking practices is two-fold: Basecamp makes it easier to adhere to a convention for tracking status because of built-in elements like milestones, and that status is easily visible to everyone. Even so, Basecamp's lack of workflow features calls into question its effectiveness as a time and effort-saving tool.

Aside from this major difference between Basecamp and the other workflow systems already discussed, there is also the issue of integration with the editing and production process. Any publication that chooses to use writeboards to edit articles must either perform substantive edits outside of Basecamp (as *Briarpatch* does); move text between the writeboard and Word whenever an article is sent to or comes back from an author; or else give the author a Basecamp login and have the author create the writeboard by importing the first draft. Each option is problematic for reasons similar to

those outlined in the Bricolage review. Similarly, Basecamp does not integrate well with layout programs at the opposite end of the editing process. Exporting text from Basecamp into InDesign is a simple copy and paste, but doing so means simple formatting may be lost depending on the layout program's support for copying and pasting from web browsers.[108]

A publication may choose to ignore writeboards entirely and instead use the generic file storage capabilities to upload Word documents. This solves the integration problems but also means users cannot take advantage of the versioning features of the writeboard. This is similar to the problem OJS faces with its file-based content management.

Software review summary

None of the four categories of software available to small magazines is a perfect fit for small magazines, though some are very workable options. All four categories contain features that should be emulated in an ideal workflow software solution, but because of major issues with each category, none are especially suitable as a plug-and-play product for small magazines.

Integrated solutions come closest to handling the various issues facing a small magazine workflow, in that they can track articles and users, handle editing and layout tasks within the scope of the system, and follow familiar publishing paradigms such as the use of industry-standard layout tools and copyfitting programs designed for a print publishing environment. However, the cost in money and computer resources of implementing such a system puts integrated solutions far out of the reach of most small

[108] As indicated in the *Briarpatch* case study, copying text from Internet Explorer to InDesign preserved simple formatting like bold and italics. However, InDesign CS2 does not recognize formatting copied from Firefox.

magazines, and the tight integration of specific layout and editing software makes it difficult for a magazine to switch to different software suites in the future.

Web-centric workflow systems like Bricolage offer a relatively accessible and easily modifiable workflow engine, but are clearly not equipped for print production in that they do not integrate well with layout applications and are biased towards XML/HTML output. Bricolage in particular also suffers from unneeded complexity in several areas, including user permissions and document structure, though these are not necessarily issues with the category in general. An academic workflow system like OJS suffers from the dual problems of hewing too closely to the requirements of an academic publishing environment, and utilizing a workflow model that is extremely hard to modify by the end user. However, OJS's e-mail communication abilities are worthy of emulation, and though its file-based content management system means editing and layout cannot be performed within the scope of OJS, it does mean that OJS will work fine regardless of what editing or layout software a magazine uses.

Finally, project management software like Basecamp suffers from a lack of specificity to the publishing environment and a general lack of workflow enforcement. This is not surprising since Basecamp was not designed to be a workflow system. However, because Basecamp is easy to use and relatively pliable in terms of how the system can be used, magazines can take advantage of Basecamp to manage editorial workflow, so long as those magazines understand that there is no workflow enforcement. Users must adhere to a set of social conventions so that, for example, editors do not edit the wrong documents or move a deadline without consulting others.

Chapter 4: Towards an ideal workflow software solution

The software packages available to a small magazine wishing to improve its editorial workflow are quite varied, but nearly all make tradeoffs regarding functionality, ease of use, and cost of adoption. Comprehensive packages that cover a magazine's needs and desires will likely cost exorbitant sums of money, effectively putting them out of reach of most small magazines. Software that integrates with layout and editing software may not be flexible enough to use in the varied and often suboptimal computing environments present at most small magazines, but web-based software may introduce errors or extra work into the editing process. Free software may require significant amounts of technical knowledge or labour to get running, hidden costs that magazines may not be able to afford. Finally, not every magazine is the same; some magazines may feel more comfortable with certain tradeoffs than others, meaning no single configuration is likely to fill every small magazine's needs perfectly.

Thus it seems that a new software solution, aimed specifically at the needs of small magazines, may be required to provide an answer where previous products failed. Whether this new software comes from repurposing or modifying an existing application or built using entirely new code from scratch, it seems clear that the ideal workflow solution will have to be constructed, not pulled off a shelf. Luckily, there are certain aspects about the workflows of small magazines that may make this task easier.

The seven magazines that participated in the study had fairly similar editorial workflows and shared certain situational traits, such as small, largely volunteer staffs and a tendency to rely on work done out of office. As a result, small magazines in general

are more likely to make the same tradeoffs and give priority to the same things when considering workflow software. In building an ideal workflow solution for small magazines, it is useful to keep in mind what priorities small magazines would choose, and thus what tradeoffs they would be willing to make.

Cost: why purchase price is just the beginning

One of the reasons why enterprise-class integrated solutions like QPS are rarely adopted or even considered by small magazines has to do with cost. The upfront cost of the software is often far beyond what a small magazine is willing or even able to pay, a fact that quickly eliminates such packages from consideration. However, the price of the software is just one part of the cost calculation any small magazine would perform, as *Briarpatch* did when it chose a software package to manage its workflow. Even free or open-source software may end up being too costly because of other costs a small magazine is unwilling or unable to pay. Thus, when considering specifications for a workflow software package aimed at small magazines, it is important to minimize as many of the various costs as possible, not simply the purchase price.

The first additional cost is computer infrastructure. Some workflow software packages require additional computers to act as servers; others require that any computer used with the software have internet connectivity or a local area network. At one extreme lies the integrated solutions; software like QPS or K4 require a local file and database server at a minimum, plus a local area network. The worst-case scenario is a small magazine that has neither, in which case equipment costs can exceed a thousand dollars. A more likely situation is that a magazine has a local network but not a file server, but since the vast majority of the equipment cost lies in the server, this is not much different from the worst-case scenario.

Web-based solutions like Bricolage and OJS incur far lower upfront costs for equipment, but require ongoing costs to be paid. Neither package requires a local server nor a local network, but simply that every client computer has access to an internet connection. For most small magazines, including all the magazines in this study, this is not a problem. However, both packages do need to be installed on a web server, and depending on the software, not just any web hosting will do. OJS only needs a server with PHP and MySQL installed, so even cheap web hosting costing less than ten dollars a month will work. Bricolage, on the other hand, requires quite a bit more, and so will be tough to get working with most cheap web hosting plans.

Finally there are hosted solutions like Basecamp that involve no setup on the magazine's part because the software is actually hosted elsewhere; in Basecamp's case, it's hosted by the developer, 37signals. Thus there is no equipment cost per se; instead, there's a monthly subscription fee for most plans (though it is theoretically possible for a magazine to utilize Basecamp's free plan).

Another potentially substantial cost is the cost of setting up such a system. This cost tends to scale with the equipment requirements each workflow system imposes on a magazine, and thus with the equipment cost. Integrated solutions require a significant amount of setup and so can be said to have the most expensive setup cost. In practice, this setup cost would likely be rolled into the sales contract any magazine negotiated with a system integrator, but it is worth noting nonetheless. Bricolage's installation procedure is fairly technical and not designed for the end user, so it requires either a technically savvy staff member or a paid freelance contractor to set it up. OJS's less onerous installation can theoretically be performed by anyone with rudimentary knowledge of web technologies, including members of the magazine's staff, so the setup

cost is only what the magazine pays for that person's time. Finally, hosted solutions like Basecamp have no setup costs at all.

A third cost is training. No matter what system a magazine eventually adopts, the staff will require a certain amount of time and effort to adapt to new methods and processes. More complex and less intuitive systems will require more training for staff to get up to speed, and while a magazine is still learning the new system its productivity will be diminished. Most of the workflow software packages reviewed have no major training options available to magazines, so for the most part only those magazines that decide on the expensive integrated solutions will be likely to pay for training. Small magazines will most likely have to spend their own time getting acquainted with whatever system they adopt, so in this sense the "training cost" is really the loss in productivity and time during the learning period. The state of documentation for each software package should be taken into account when determining ease of adoption, especially for software that comes without formal training options or magazines unable to take advantage of training. For example, Basecamp offers video tutorials of all main features, and OJS offers a number of tutorials such as the "OJS in an Hour" guide available from the OJS website. Support communities made up of end-users and developers may also be assets that partially offset the lack of training.

An ideal workflow solution will likely be a web-based solution, either hosted by the magazine itself on a web server or else by a central service that would charge magazines a nominal fee. A web-based solution is preferable because though self-maintained and hosted options involve a small regular fee, neither requires very much in the way of computer infrastructure compared to non web-based options. Locally hosted web solutions can be installed fairly easily if designed for it, as OJS was, and service-based solutions like Basecamp require no installation at all. Finally, because small

magazines are unlikely to undergo a significant amount of training before adopting a workflow solution, designing software with usability in mind, especially usability for novice computer users, is a must. Any workflow solution with a user interface that is difficult to understand and hard to use will be deemed useless and discarded fairly quickly, even if the software is otherwise more than capable of handling the magazine's needs.

Specialized versus general-purpose software

Basecamp stands apart from all the other software packages reviewed in that it was not built with any sort of publishing environment in mind, nor was it designed as a workflow tool. Considering that Basecamp does not manage editorial workflow in the sense that the other packages reviewed in this report do, it may not be obvious why a publication would consider or adopt Basecamp versus one of the other packages. However, of the seven magazines participating in the study, only two adopted a software package for the purposes of managing their editorial workflow, and both magazines chose packages that, strictly speaking, have no workflow features to them. *Briarpatch* chose Basecamp after researching other options, and *Spacing* adopted phpBB, a web-based discussion board system, right from the start.

Briarpatch chose Basecamp as a workflow management solution for several reasons. It was cheap, unlike more advanced solutions that integrated with InDesign and InCopy. It was easy to set up and maintain, unlike Bricolage, which required a lot of technical expertise to install. Finally, its feature set was robust enough to handle *Briarpatch*'s editing and organizational needs, unlike Google Documents. Basecamp, then, was the best available solution given the various options available. But does that mean general-purpose applications like Basecamp and phpBB are good solutions, or merely least worst solutions?

Ricepaper is a Vancouver-based quarterly magazine devoted to the Asian-Canadian literary scene and associated community. In 2005, then editor-in-chief Jessica Chan published a report detailing the magazine's attempt to improve its operations by developing a content management system, a project it called Ricecooker. The Ricecooker CMS went through several iterations over the course of three years, and despite the input of the magazine's entire staff and the efforts of several development teams over the years, the Ricecooker CMS still had not reached a stable state of development by the time of the report. Rather, the system had undergone periodic reinvention and refinement in an effort to cure the perceived deficiencies of each previous version. In what was becoming an increasingly predictable cycle, however, each new version of the Ricecooker CMS produced a new set of problems:

> The Users Committee, which was first formed in November 2002 to conduct tests on Ricecooker 1.0, had spent almost two years on testing and researching requirements for the CMS. At an estimated eight hours a week, this meant that the Users Committee had poured about 640 hours into the system thus far. While the staff had consistently produced the magazine alongside their development work, the senior management questioned if they had already spent too much time and effort on the project. In fact, the Publishers even suggested dropping the CMS entirely, especially since no one was actually using Ricecooker 3.0.[109]

Ricepaper decided to work on a fourth version of the Ricecooker CMS. In the meantime, however, the magazine needed a system to serve as an interim solution. The magazine's IT staff created Ricepot in June 2004 by installing phpBB on a web server. Initially intended simply to serve as a method of facilitating online communications between staff, the various departments of the magazine began to migrate all their operations to the forum over time, including tracking the workflow status of articles and storing documents. *Ricepaper*'s experiences with both a custom-made content

[109] Jessica Chan, *Where the Rice Cooks: Connecting Text and Community at* Ricepaper *Magazine Through Developing a Content Management System* (Vancouver, Simon Fraser University, 2005), 58.

management solution, specifically designed to fit the needs of a magazine publishing environment (with varying degrees of success), and a general-purpose discussion forum package, offers a possible case for utilizing general-purpose software like Basecamp over more specialized solutions.

Chan described the assessment of both the Ricecooker CMS and the Ricepot discussion board as demonstrative of the magazine's true needs versus its perceived needs in the eyes of the team leading the development of the Ricecooker CMS:

> On the surface, Ricecooker 3.0 would appear to be an ideal system for a volunteer-based magazine: the system could automatically generate updates and cross-reference data, thereby cutting the labour spent on compiling update reports. Upon further consideration though, this was not a critical need for *Ricepaper*. In fact, weren't the benefits of Ricepot an indication of what the magazine really needed—an expandable system that was simple to administrate, and facilitated transparent communications?[110]

Chan also noted that while the Ricecooker CMS was theoretically designed with *Ricepaper*'s workflow issues in mind, staff members had to adapt to the CMS's way of doing things just as much as they did with the Ricepot discussion board, despite the fact that the discussion board was not designed to manage workflow at all. The conflict at *Ricepaper* lay in choosing between a general-purpose solution that was not ideal but appeared to be workable; an existing specialized workflow solution that was largely unworkable; and an ideal specialized workflow solution that would solve all problems but did not actually exist and perhaps never would. Of those three options, Chan concluded that the general-purpose solution should not be dismissed out of hand, and deserved a closer look as a permanent solution.

[110] Chan, *Where the Rice Cooks*, 71.

Assuming, however, that an ideal specialized workflow solution existed, and a magazine had to choose between the general-purpose solution and the specialized solution, there are still good arguments that favour the general-purpose option. Taking Open Journal Systems (OJS) as an example, it is possible for a specialized workflow solution to be too rigid in its workflow model; because OJS is designed specifically for academic journals, it works exceedingly well in that environment but few others. By tailoring its feature set so closely to the needs of one specific type of publication, OJS ends up being harder to adopt than a general-purpose solution like Basecamp that makes no such assumptions; flexibility wins over specificity. Along the same lines, a workflow management system like Ricecooker, built specifically to *Ricepaper*'s specifications, may not work as well at another magazine with a different staffing situation and editorial process.

The goal, then, is to create a workflow solution that is sufficiently flexible to handle the full range of small magazine environments, while providing just enough structure to facilitate the sorts of interactions all small magazines deal with. For example, because all magazines perform roughly the same editing tasks, it would be ideal for a workflow solution to provide a comprehensive editing tool that allows for annotations and versioning control. However, because magazines may assign different numbers of editors to the various editing stages, or shuffle those stages around, it would be better if a workflow solution allowed the user to define those aspects of the workflow model instead of enforcing one specific model on all the magazines using the software.

Integration with pre-existing editing and layout applications

The workflow software packages available to the magazine industry range from packages that make no assumptions about the editing and layout software a magazine

uses to packages that enforce a specific set of software for every aspect of the editing and production process. In other words, the question of integrating the workflow software with editing and layout programs is a specific instance of the "specialized versus general-purpose" argument above. Should a workflow solution enforce a specific text editor and layout program, or is it better for a workflow solution to leave those options up to the magazine? Ideally, given the exact same functionality, it would be preferable for a workflow solution to give users the choice of what editing and layout software to adopt.

The reality, however, is that giving users a choice means losing certain functionality that can only be gained from tight integration. One of the advantages of integrated solutions is that it is easy to access the latest edit of an article at all times, even if the article is in layout. This means art directors and designers can start creating a layout around the first draft of an article and keep up to date with that article as it passes through the various stages of editing. This is only possible because integrated solutions are hooked into a specific suite of programs—for example, Adobe InCopy and InDesign for Softcare's K4. Introducing new programs to the mix would require a significant amount of work on the part of the workflow software developer to add support for that program; Quark and Softcare have added support for Microsoft Word to newer versions of their integrated solutions, for example.

In contrast, other solutions like Bricolage make no attempt to integrate with any editing or layout software. Bricolage provides its own editing tools and leaves the question of layout and production largely out of the system, while OJS relies on files to hold both article text and layout. OJS's approach works with whatever editing and layout software a magazine has, but it also precludes the possibility of the parallel editing and layout workflows seen in integrated solutions. Bricolage largely relies on its own tools for editing, requiring that editors import and export text directly into Bricolage, and

similarly exports a final product for use in other applications (or web templates, as Bricolage was originally designed to do). Neither option is as elegant as the parallel editor-designer workflow.

But there are significant downsides to an integrated workflow as well. Any magazine that adopts an integrated solution is locked into a specific software ecosystem. With QPS, a magazine must use QuarkXPress for layout; with K4, Adobe InDesign. Older versions of both products also require that you use QuarkCopyDesk and Adobe InCopy respectively, since integration with Microsoft Word was added relatively recently to both workflow solutions. Any other software a magazine might use for editing or layout must be thrown out, and any mandated software a magazine does not yet own must be purchased. Additionally, by locking a magazine into a particular set of software, integrated solutions make it far more difficult for a magazine to adopt new software, or even new versions of software, without either updating or changing the integrated solution. One case where such integration can be costly is the recent industry shift from QuarkXPress towards InDesign as a layout tool. Because of major feature additions and improvements to InDesign, many magazines have switched to using it over QuarkXPress. Such a move would be far more costly if it also meant a magazine had to change its integrated workflow software as well.

Issues with publishing content to print and online media

Ideally, any workflow management system would be able to publish content to multiple media reliably and easily—a concept often called single-source publishing. Single-source publishing assumes that the content of a publication is independent of the finished form it eventually takes, whether it be a print publication, a website, or some other physical or digital form. As such, systems that utilize the concept of single-source

publishing treat content as media-agnostic until the final stage of the production cycle, when it comes time to publish content to different media.

There are two major advantages to such a system for small magazines looking to publish content to a website as well as to traditional print issues. The first is ease of use. Several of the magazines in this study have websites with republished content from back issues, but none had any automated system for putting those articles up on the website. Jessica Johnston of *This Magazine* was not completely clear on the specifics of how articles were posted to the website—a common trait amongst the editors who responded to the study—but said the process was "fairly labour intensive" and was not automated.[111] In general, the process of publishing to the web appears to be similar across magazines: after the layout files for the print publication are complete, the person handling web publishing duties extracts the final text from the layouts (either PDFs or InDesign/Quark layout files), collects whatever imagery goes along with the article, wrangles the content into whatever content management system the magazine has set up for the site (or, in some cases, raw HTML templates), and posts the final product to the website. This manual process is error-prone and tedious, making any sort of automated publishing process very attractive.

The second advantage of a single-source publishing system is consistency across media. One of the ways the manual process can fail is if the web manager receives the wrong layout files or, worse yet, has to rely on versions of the text that have not been laid out and thus miss whatever copyediting and proofreading changes were made after the copy entered layout. For example, Aaron Spitzer of *Up Here* noted that articles on their website were based on copy taken from the Word documents, meaning all changes made

[111] Johnston, phone interview.

to the copy after the layout stage never made it to the website.[112] Theoretically, single-source publishing would prevent this sort of problem from occurring.

The potential for single-source publishing to improve a magazine's workflow at the tail end of the process is clear, especially if the magazine has a significant web presence. However, none of the workflow software solutions reviewed incorporated any significant single-source publishing features that encompassed both print and web media. Any developer that wishes to create a system that publishes to both print and web has to overcome several problems first.

Representing content: files or text?

Of the web-based workflow solutions reviewed, there was a split between using files to hold content and storing content directly as editable text. OJS does the former, while Bricolage does the latter; Basecamp allows for both approaches. As the critiques of each program showed, both approaches have advantages and disadvantages. By storing the text of articles directly, a workflow solution can repurpose that text for other media quite easily. Bricolage, for example, was designed for output to a variety of different web-centric formats, including PHP and several Perl-based XML/HTML template languages. With most file-based approaches, the actual text would be unavailable to the system; OJS cannot automatically publish to any media at all, and relies on the layout editor to provide the relevant HTML and PDF output instead. However, the file-based approach has the advantage of not having to worry about the data inside the files, while the text-based approach must provide rich text editing tools for maintaining and editing the text. *Briarpatch* and *Spacing* both noted that the quality of the text editing tools in Basecamp and phpBB was fairly poor, especially when it came to maintaining simple formatting like bold and italics in the text. Furthermore, both magazines had import and export

[112] Spitzer, e-mailed questionnaire.

issues to deal with, both when sending the text of an article back to an author and when importing that text into layout software like InDesign. Single-source publishing would theoretically solve the latter issue, but not the former.

Clearly, if single-source publishing is a priority for a workflow software solution, the text-based approach is the only workable method. However, additional engineering effort will be needed to create the rich editing and author collaboration tools that small magazines need during the editorial process, or else the workflow software will not be able to completely encompass a magazine's needs.

Publishing to print without an integrated workflow solution

Any workflow solution that attempts to publish to print without integration with a layout application like InDesign will have to solve the issues of exporting to and managing content in the layout application. Because layout applications have not included robust importing tools until relatively recently, most layout processes still use the traditional method of importing text straight from a text file or a supported word processing format like Word documents. Not only do most small magazines, including all of the magazines in this study, ignore the presence of XML features in both InDesign and Quark, they also ignore the possibility of importing text from a database.

One look at the documentation of XML functionality in layout applications and consideration of the general level of technical expertise of most editors and designers, and it becomes clearer as to why most small magazines have largely ignored XML. Even assuming editors could manage text in an XML format, the methods of importing XML content into a layout document require designers and art directors to learn how XML works. On the editing side, only recent versions of Word have any XML functionality at all, and the XML files they output are practically unreadable by humans (and thus

extremely difficult to utilize in InDesign).[113] Asking editors to produce and edit raw XML themselves is a high order, especially given that most of the editors interviewed admitted their comfort level with technology was average at best.

Workflow software that produced XML output as a means of sending content to a layout application would solve the problem of requiring editors to deal with XML, because editors would simply deal with the built-in text editing facilities of the workflow software instead of having to edit raw XML. However, the issue with setting up the layout application to take XML input is still present, and not easily solvable by magazine staffs. There are other problems as well—because most small magazines do significant editing work after an article has been placed in layout, the problem of how to incorporate those changes into both the workflow software and the layout application arises. One option is to propagate edits from the workflow software. Changes would have to be made in the workflow software, re-outputted to XML, and re-imported into the layout software, a relatively tedious process compared to the traditional method of simply making changes in the layout software.

Another option is to modify or extend the layout software so that it could pull data from and push data to the workflow software database. The major advantage to database integration is similar to the advantage to using an integrated solution: changes to the text in the database can propagate to the layout software relatively easily, and changes made in the layout software can propagate back to the workflow software database. The major disadvantage, however, is neither QuarkXPress nor InDesign have support for handling databases built in, and most solutions for getting database data into layout software involve publishing the contents of the database to a flat file first, and importing the file—in other words, the same process as an XML-based publishing

[113] For a particularly hideous example, see Listing 1 at http://www.devx.com/dotnet/Article/17358?trk=DXRSS_XML (accessed June 4, 2008).

solution. There are solutions that involve direct connections to a database—em Software's InCatalog Pro and Cacidi System's LiveMerge are two products that allow InDesign to interact with SQL databases. However, InCatalog Pro costs $1200 USD per install[114] and LiveMerge costs $1000 USD per install[115]. Both packages may be too expensive for a small magazine, especially considering the other potential costs associated with adopting a workflow software solution.

The solution most likely to be compatible with the resources of small magazines is an XML-based solution, but any such solution will require significant changes to how publications import data into their layout software of choice. Given the reduced impact on a magazine's workflow, a low-cost database plugin for InDesign and Quark would also be a big step towards integrating a workflow software package into a print production environment.

[114] Em Software, "InCatalog," http://www.emsoftwarestore.com/incatalog.html (accessed June 4, 2008).

[115] Cacidi Systems, "Webstore," http://www.cacidi.com/webstore.php?lang=UK&jobID=PROD (accessed June 4, 2008).

Chapter 5: Conclusion

Creating a comprehensive software package aimed at small magazines is a daunting prospect. The software has to be sufficiently comprehensive that it can facilitate a large enough portion of a magazine's workflow to be useful. It has to be flexible enough to handle a wide variety of workflow models. And it has to be easy to install, set up and use so that magazines don't prematurely abandon the software. Any developer attempting to do this must also deal with the reality that small magazines have very little in the way of resources, meaning the software will have to run on fairly marginal computing environments, and the final cost of the program must be fairly low or else the product will be priced well out of the target market's range.

There is also the issue of how a small magazine will adopt new software that could radically change their current workflow practices. Jeffrey Veen of Adaptive Path, a usability consulting firm, noted that people tend to stick to the practices they have experience with, even if those practices are sub-optimal:

> Knowledge workers spend years building strategies to accomplish their jobs, practices that likely date back to study skills acquired during their education. So changing those processes—no matter how valid the provided technical solution—is nearly impossible. Users will rebel, even after substantial training.[116]

Given all these constraints, it may be too much to expect to see an ideal editorial workflow software solution in the near future—there are simply too many roadblocks and not enough incentives for experienced developers. However, by using alternative

[116] Jeffrey Veen, "Why Content Management Fails," Adaptive Path website, http://www.adaptivepath.com/ideas/essays/archives/000315.php (accessed July 17, 2008).

forms of development like open-source development, utilizing already existing code and functionality from other open-source projects, and using an iterative development process to slowly build the product one piece at a time, it is possible to start building an editorial workflow solution that begins to address the issues small magazines face, and build in additional functionality as time and resources allow.

Starting from scratch versus using existing code

Because there are several open-source projects available that could theoretically be modified to better fit the needs of small magazines, the question arises of whether it would be better to take advantage of the work poured into these projects and create a branch of an existing codebase, or to start from scratch.

Ideally, the major advantage to reusing code from a previous project is being able to take advantage of other people's work in order to shorten development and utilize tried-and-tested code. Whether this advantage is real or imagined depends greatly on the codebase being cannibalized. If the original program code is poorly documented, badly structured, or just plain unusable, then modifying it may be significantly more difficult than program code that was designed with extensibility and modification in mind. In all cases, there is a period and a great deal of effort that must go into deciphering and fully understanding the codebase before substantial modifications can be made.

Another less obvious advantage to code reuse is that assuming a large enough userbase, the modified project may be able to take advantage of an already existing developer community. This becomes less true the more a product is modified, however, as the existing developer community for the original product becomes less familiar with the offshoot project. Nevertheless, this can be a substantial asset if parts of that developer community are interested in the offshoot project; a good example is the

development of the Mozilla Firefox browser, which started as an offshoot of the Mozilla suite.

There are several crucial advantages to using all-new code, however. The first is that by starting from scratch, the software can be built to specification rather easily compared to repurposing existing code. Trying to change old code requires removing unnecessary features, adding new ones in, and hoping everything plays nicely once complete. It also requires tracking the effects of removing old code and adding new code; heavy modifications can cause major instability if not done properly. Starting from scratch, on the other hand, means knowing from the beginning exactly what the code does and what will happen if it is changed. It is more likely to remain stable throughout development and more likely to include only what code is needed and leave out unnecessary code.

Another major advantage is that new projects can utilize new technologies and methods more easily. For example, if future development on a workflow solution points to a web-based solution, as implied in the previous chapter, a variety of web frameworks can be used that define a common vernacular for many basic operations such as accessing databases, creating web forms, displaying page templates, and more. These frameworks are fairly new, and represent a leap in sophistication over previous methods. Repurposing old projects, then, involve either creating new code in the old style, or else refactoring the old project to use new technologies. In the latter case, a major refactoring may involve just as much work as starting from scratch, because in essence it *is* starting from scratch—redefining the basic operations of a program and coming up with new, cleaner code to perform those operations.

The value of open-source development

Bricolage and OJS are both built on open-source code, as are many other web-based content management systems. Because their permissive licenses allow for the free modification and redistribution of the codebase, it is theoretically possible to pick a software package that already does much of what small magazines require and modify them as necessary. Such a development model may be preferable to the more traditional proprietary development model, where a company builds a software package from scratch or using proprietary code as a basis, for several reasons in addition to the ability to modify pre-existing code.

First, as explained above, the prospect of funding software development through sales to small magazines may be a difficult one because small magazines traditionally have little to no budget set aside for software improvements. Thus the incentive to traditional software developers is low, barring the intervention of government programs or magazine associations that can pool resources to fund development. Second, because traditional developers rely on a proprietary codebase, additions to the code can only be made by people the developer trusts, such as its own employees or contractors. Open-source projects, on the other hand, allow anyone the possibility of contributing code, meaning a much greater number of programmers can be brought to bear on a project in theory. Third, because the resulting code will be open-source, there will never be any issues with a small magazine being unable to customize software to their needs just because of a restrictive software license or a lack of source code; the code will always be freely available. Finally, because the code is freely available and owned by no one, it will always theoretically be in development. The product will never disappear from the market because a company discontinues development, and so long as there is interest in the program's continued refinement, there will always be new releases.

This report will not go into a detailed comparison of open-source versus traditional development models; it will suffice to say that though open-source development is not a perfect solution, it is far more likely to produce an actual product than relying on traditional, market-driven development models to provide for the small magazine industry when it has failed to do so for the past two decades.

Iterative development: produce now, refine later

The basic concept behind an iterative development model is simple: prioritize the planned functionality of the software; build the most important features first; release a first version with basic functionality; refine current functionality and add new functionality over time.[117] There are several advantages to iterative development. First, it can speed up the delivery of software. As Jessica Chan's report on the Ricecooker CMS shows, given the choice between a basic software package that can be delivered now and an ideal software package that only exists in theory, the basic software package is preferable.

Chan's report provides another reason for utilizing an iterative model: by starting from scratch each time *Ricepaper* rebuilt its CMS, the magazine continued to create complex systems that often did not work to specifications and missed key details about how the editorial process worked at the magazine. Theoretically, an iterative process would solve these problems by continually integrating user and developer feedback into software refinements and additions. Finally, iterative development can be more productive in a development environment with particularly limited resources. In the worst-case scenario, should development ever cease on the software, at least there will

[117] For a more thorough explanation of the iterative development model and how it was created, try Václav Rajlich, "Changing the paradigm of software engineering," *Communications of the ACM* 49, no. 8 (August 2006) : 67-70. Books that discuss iterative development include *Extreme Programming Explained* by Kent Beck and *The Mythical Man-Month* by Frederick P. Brooks Jr.

still exist a “good-enough” version that magazines can continue to use. This is far less likely when the first version of a product has to encompass all the planned functionality.

So what should be prioritized for development and what can be left to later efforts? A basic mechanism for tracking articles is the obvious first step, as even without the ability to store article text or edit articles within the software, having a way of tracking the status and possession of articles that can be accessed by all staff members provides real value. The cases of *Briarpatch*, *Spacing* and *Ricepaper* provide real-world examples of successful systems that do little besides track articles and users in the workflow. Subsequent development should focus on making the primary task of process tracking and communication easier, by adding additional methods of communication like discussion boards, and adding rigour to the tracking process by bringing article texts into the system and providing basic editing functionality, version tracking, and other article-related features. At this point the software would look much like OJS or Bricolage, but aimed specifically at a small magazine audience and stripped of any layout or print production capabilities. Development past this point could concentrate on a number of areas, such as integration with print layout software and advanced editing tools like commenting and tracking textual changes. There should also be a continued focus on refining the user interface in particular, making it simpler and easier to use with every revision even as the software itself grows more complex.

Finding the architects

Who would build such a system? Traditional developers are unlikely to take on such a project for the reasons listed above. Individual small magazines are unlikely to take on the development of a system themselves; even *Ricepaper*’s fairly ambitious project lay in limbo as of the writing of Chan’s report, and most magazines don’t have a volunteer IT staff to assist with development. Larger magazines could theoretically build

such a system if their budget was large enough, and then small magazines could rely on a trickle-down effect to eventually gain access. The problem is that even fairly large multi-title publishers may not have the resources or the will for such an undertaking; my experience at *Toronto Life* shows that even large Canadian magazines work without the benefit of workflow software. And once a publisher reaches a certain size, it may be in their best interests to simply purchase a pre-existing solution instead of developing a new one. Even if a large publisher does commission new development, there is no guarantee that small magazines will ever have access to the results, let alone that the results will fit their needs versus those of the large publisher.

This leaves very few interested parties. Small magazine associations may be able to pool the resources of their members to fund development, or interested third parties like universities or government-assisted programs could provide the resources. A combination of the two options is probably the most likely chance for possible development, but it is worth noting that this is unlikely to be a market-based solution.

First steps

This report is merely a first inquiry into the subject of editorial workflow software in a small magazine environment. A wider survey of small magazines would provide a greater body of research on which to base a potential set of requirements. More examination into magazines' attitudes towards the adoption of software and their comfort level with technology in general and new software specifically would be particularly helpful in determining the scope of potential software, as well as many specifics regarding functionality and interface. An examination of the available technical resources available to small magazines would also be helpful in sorting out what the magazines themselves can provide in terms of programmers and testers; if the main

development thrust can be provided by the small magazines themselves, the software would be in better hands than if development were led by a less interested third party.

Though this report examines the functionality and interfaces of various software packages, it does not review the underlying code of any of those packages. Therefore, if code reuse is made a priority, an examination of the open-source projects reviewed here, as well as others in the same categories, will be required to determine what would be a suitable base for redevelopment—not simply in terms of desired functionality, but also in terms of how flexible and extensible the underlying codebase is. However, in may in fact be better to instead start from scratch and create a new application specifically designed with small magazines in mind. Such an approach would avoid the costly exercise of triaging currently existing code to pick a candidate, then thoroughly dissecting it to figure out how to modify it to specifications.

It is helpful to remember that even the ideal software solution is possible, technologically speaking. If there is a roadblock to the ideal, it lies not in technical ability, but in the ability of the small magazine industry to provide for and guide development of such software. This is partially because of the structural issues facing small magazines in general, and partially because of a lack of experience dealing with software development. Hopefully, this report will serve as a guide for small magazines and interested developers such that they can work though these issues, so that eventually all small magazines can reap the benefits of a more efficient and reliable workflow system.

Appendices

Appendix A: Example step-by-step workflows for reviewed software

For those looking for a more detailed description of the software packages reviewed in Chapter 3, here are step-by-step workflows that explain in greater detail how an article would move from first draft to final product in each software package. Some software may allow for different approaches, in which case the example workflow will reflect one possible method of structuring the editorial workflow.

Smart Connection Pro

This example workflow assumes a text-driven workflow, as opposed to a layout-driven workflow. Alice is the handling editor and Bob is the art director handling layout duties. There are also four baskets: the Edit basket, the Copyedit basket, the Proofing basket, and the Finals basket. For simplicity's sake, this workflow also assumes that the layout contains just one article; the workflow becomes slightly more complex for the art director if an InDesign file contains multiple articles.

1. Alice imports the original article into InCopy and saves it to the proper issue, section and basket—in this case, the Edit basket. The article is now available for Bob's use.
2. Alice continues to make edits in InCopy and corresponds with the author.
3. Meanwhile, Bob creates a new InDesign layout, saving it to the proper issue, section and basket (again the Edit basket). He then imports the article into InDesign using the Smart Connection palette. Bob can now design a layout with the text of the article already present. Bob cannot make changes to the text but can change layout parameters like text formatting styles and the dimensions of the text box.

4. As Alice receives changes from the author, she checks out the article from the Smart Connection system and merges the changes. She saves and closes the document after each round of changes, which automatically checks the article back into the system.
5. Bob receives notification of Alice's changes, and can choose to update the text at any time. Similarly, as Bob makes and saves changes to the layout, Alice receives notification of any updates (if she has the article checked out at the time) and can choose to update the layout in InCopy for copyfitting purposes. Otherwise, Alice and Bob work independently of one another, even though both are working with each other's content.
6. Once Alice is happy with the article, she sends the article to the next basket in the workflow, the Copyedit basket.
7. The copyeditor makes changes and sends the article to the Proofing basket, indicating the article text is ready for proofing and copyfitting.
8. Once Bob has finalized the layout, he moves the layout to the Proofing basket as well, indicating that the layout is ready for copyfitting.
9. After the article has been copyfit and proofed in layout, both the layout and article move to the Finals basket to indicate work has been finished on the article.

Bricolage

In this example, an article has Alice as the primary handling editor and Bob as the secondary editor. In this workflow, there are four desks: one for Alice, one for Bob, one for copyeditors, and one for proofing and publishing. A publication can modify the desk setup to meet their particular needs—for example, it may not be necessary to give each editor their own desk, or for a publication with multiple copyeditors it may be better to give each copyeditor their own desk.

1. Alice creates the Story element in Bricolage and sends it to her desk for editing.
2. Once Alice is finished, she checks the story back into the system and moves the story to Bob's desk for further editing.
3. Bob checks out the article and begins editing it. If everything is normal, he makes his edits and sends the article to the copy desk after he checks it in. This time, however, he notices something in the article that requires a major change. After

consulting with Alice, he checks the article back into Bricolage and sends it back to Alice's desk.

4. Alice consults with the author via e-mail and the author sends back a new draft. Alice then merges the changes with the Bricolage document and checks it back into the system. Bob has already made his changes so the article goes on to the copy desk.
5. The copyeditor checks out the article in the copy desk, makes edits, checks the article back in, and moves it to the proofing/publish desk.
6. Using Bricolage's preview function, all the editors proofread the article and make changes individually. Finally, the editor-in-chief, Carol, okays the article for layout and publishes the article to a flat file.
7. The art director takes the flat file and imports it into a layout program. The rest of the workflow takes place outside of Bricolage; ideally, all proofreading has been done at this point.

Open Journal Systems

The workflow is defined in terms of roles (please see the "Overall workflow structure" section of the OJS review in Chapter 3 for more information on roles), so this example need only define the people taking part: Alice is the author, Bob is the journal's editor, Carol is the section editor, David is the copyeditor and Esther is the layout editor. The magazine has no proofreaders, so Carol will perform the proofreading duties.

1. Alice prepares the article for submission by logging into OJS and following the five-step submission process, in which the author enters the necessary metadata and uploads the original Word document. The completed submission triggers an e-mail to the editor.
2. Bob logs into OJS and notices there is an article in the submission queue. He assigns the article to Carol.
3. Carol receives notification of the assignment and checks her review queue for the article. She downloads the Word document and makes her edits, then uploads the new file and marks the "Revisions Required" option in the Editor Decision section. She then uses the Notify Author link to send an e-mail to Alice.

4. Alice downloads the revised file, makes her edits, and uploads a new draft to OJS. Carol and Alice continue to trade files in this manner until the article is in a satisfactory state.
5. Carol marks the final draft as accepted in the Editor Decision section of the Review pane, and selects the final draft file to send to copyediting. She then moves to the Editing pane and assigns David to perform the copyediting. She sends David a request e-mail through OJS to start the copyedit.
6. David reviews the document, makes his changes, and uploads a new file. Once he is finished with the file, OJS alerts Alice that her article is ready for her copyedit.
7. Alice copyedits the article, uploads a new version of the file if necessary, and sends it back to David.
8. David does a final copyedit and uploads a new version if necessary.
9. Carol receives notification that David has finished the copyediting process and assigns Esther to put the article into layout. She re-uploads the final copyedit version of the file and signals Esther using the Request link.
10. Esther imports the article into a layout program and begins designing the pages. Once she's done, she uploads the PDF to OJS.
11. Carol starts the proofreading stage by requesting a final read from Alice.
12. Alice performs the author proofread and uploads her changes.
13. Carol initiates the proofread in the Editing pane, downloads the author proofread file, and uploads her changes. She then signals Esther that final changes need to be made to the layout.
14. Esther downloads the final article document, fixes the layout as necessary, then uploads the final PDF.
15. Carol schedules the article for publication in the next issue. The article is now complete.
16. Once all the articles are ready to go, Bob finds the issue in the Future Issues list and clicks the Publish Issue button.

Basecamp

Because Basecamp allows for a very broad variety of workflows that are just as much defined by a magazine's practices outside the software, it's difficult to give an example workflow that could be considered comprehensive. Instead this review will

explore how the various Basecamp tools work, such that readers can get a better idea of how the tools work individually and with each other. The Basecamp website also contains comprehensive demo videos of each tool. Finally, the *Briarpatch* case study offers another example workflow. Because the *Briarpatch* example demonstrates the use of writeboards to hold article text, this example will instead utilize Basecamp's file storage capabilities to demonstrate a different method of maintaining content. Alice is the handling editor, Bob is the copyeditor, Carol is the layout editor, and David and Esther are proofreaders. Frank is the editor of the magazine and the administrator of the Basecamp instance.

1. Having planned out the article lineup for the next issue, Frank creates a new Basecamp project for the issue. He creates several milestones that represent deadlines for each stage of the editing process: all substantive editing done in four weeks, all copyediting in six, all layouts in seven, and all proofreading in eight, with the press deadline set for nine weeks from now.
2. Frank also creates to-do lists for each article that indicate who is assigned to each stage of the process. For the specific article in this example, Frank has created a to-do list with a substantive edit item assigned to Alice, a copyedit item assigned to Bob, a layout item assigned to Carol, and proofreading items assigned to David and Esther. Other articles may have different people assigned to each item in the list. Frank also creates a final checklist for the whole issue: a final layout item for Carol, and a final read item assigned to himself.
3. Alice receives the article from the author in Word format, renames the file so it has a descriptive title, and uploads the file to Basecamp. She gives it a category corresponding to the relevant section of the magazine for organizational purposes, and in the upload description she puts in the title of the article, and a short status message: "first draft."
4. Alice edits the article in Word and trades drafts with the author. As she sends out drafts and receives new ones back, she uploads them to Basecamp using the "upload new version" link attached to the original file she uploaded. That way, new versions will stack on top of the old file, allowing others to see and download previous drafts if they wish.

5. Once Alice is happy with the article, she uploads a new version and changes the description's status message to "copyedit needed." She also ticks Bob's name on the e-mail notifications list, so that when the upload is complete, Basecamp will send Bob a message about the new file. Finally, she checks off the "substantive edit" checkbox in the article's to-do list.
6. Frank checks the to-do lists on a regular basis to see when people have filed drafts that are ready for copyediting. Once all the substantive editing tasks have been checked off by the various editors, Frank checks off the substantive editing milestone, indicating that the milestone is complete.
7. Bob downloads the latest version of the article that Alice uploaded and starts his copyedit. One day before copyedits are due, Bob logs into the Basecamp instance and sees in the project overview calendar that the copyediting deadline is close. He works double-time to finish all his work and uploads a new version of the article, changing the description to "copyedit done, ready for layout" and ticking Carol's name on the e-mail notification list. He then checks off the copyediting item on the article's to-do list.
8. Once again, Frank waits for all the copyeditors to check off their respective copyediting items in the article to-do lists. Frank can see that while Bob has filed all his stories, some of the other copyeditors are slightly behind; this causes the milestone to appear red on his project overview calendar. Finally the copyeditors file their stories and Frank checks off the copyediting milestone two days behind schedule.
9. Carol downloads the article and drops it into layout. As she places articles into layout, she checks off the relevant item in the article to-do list. When she's ready to produce proofs, she uploads PDFs of the pages as she finishes them, ticking David and Esther on the e-mail notifications list for each file.
10. David prints out his proofs and edits them on paper, while Esther uses Adobe Acrobat to mark up her pages electronically. Esther uploads the new files to Basecamp using the "upload new version" link, while David gives Carol his marked pages in person at the next editorial meeting. After this is done, both proofreaders check off their items in the article to-do lists.
11. Carol merges the changes and uploads a new set of pages, now as one PDF, for Frank's approval, ticking Frank's name on the e-mail notifications list. She checks off the final layout item in the overall issue's final checklist.

12. Frank reads through the final issue and makes the final changes on paper. He gives the printouts to Carol to merge the changes, and she sends back a final PDF. Once Frank is satisfied, he checks off the final read item on the checklist. When Carol has sent off the PDF to the printers, Frank checks off the press deadline milestone.

Reference list

37signals. “Basecamp example uses and case studies.” http://www.basecamphq.com/examples.

Cacidi Systems. “Webstore.” http://www.cacidi.com/webstore.php?lang=UK&jobID=PRO.

Chan, Jessica. *Where the Rice Cooks: Connecting Text and Community at* Ricepaper *Magazine Through Developing a Content Management System*. Master of Publishing Internship Report, Simon Fraser University, 2005.

Em Software. “InCatalog.” http://www.emsoftwarestore.com/incatalog.html.

Fiore, Marrecca. “Publishers Turn to Content Management Systems to Increase Productivity and Profitability.” *Folio*, September 19, 2006. http://www.foliomag.com/2006/publishers-turn-content-management-systems-increase-productivity-and-profitability/.

Kinsman, Matt. “How Technology is Transforming Magazine Publishing.” *Folio*, June 30, 2006. http://www.foliomag.com/2006/how-technology-transforming-magazine-publishing/.

Manjoo, Farhad. “The next Web revolution.” *Salon*, August 10, 2005. http://dir.salon.com/story/tech/feature/2005/08/10/37signals/index1.html.

Maxwell, John W. “Extending OJS into small magazines: The OMMM Project.” *First Monday* 12, no. 1 (October 1, 2007). http://www.uic.edu/htbin/cgiwrap/bin/ojs/index.php/fm/article/view/1962/1839.

Quark, Inc. “Quark Publishing System 7 System Requirements.” http://www.quark.com/products/enterprise/modules/qps/tech_info/sys_req.html.

Quark, Inc. “Quark Publishing System: How it works.” http://www.quark.com/products/enterprise/modules/qps/pdf/QPS7_HowItWorks_US_Web.pdf.

Silber, Tony. “Switching Edit/Design Platforms.” *Folio*, August 2, 2007. http://www.foliomag.com/2007/switching-edit-design-platforms/.

Softcare GmbH. “Softcare home page.” http://www.softcare.de/.

Softcare GmbH. “Softcare K4.” http://www.softcare.de/publishing-solutions/softcare-k4/index.html.

Veen, Jeffrey. “Why Content Management Fails.” *Adaptive Path website*. http://www.adaptivepath.com/ideas/essays/archives/000315.php.

Vlietinck, Erik. “IT Enquirer Report on Editorial Workflow Systems.” *IT Enquirer*, 2007. Available via http://www.it-enquirer.com/main/ite/more/qps_sce_k4_report/.

Wheeler, David. “Content Management with Bricolage.” *O'Reilly perl.com*, August 27, 2004. http://www.perl.com/pub/a/2004/08/27/bricolage.html.

WoodWing Software. “Express Newspapers in the UK selects WoodWing.” http://woodwing.com/en/Newsflashes/20071218_Express_Newspapers.

WoodWing Software. “New York Sun Daily Newspaper Chooses WoodWing.” http://woodwing.com/en/Newsflashes/20080418_New_York_Sun.

WoodWing Software. “Smart Connection Enterprise Brochure.” http://www.woodwing.com/userfiles/file/Brochures/Smart%20Connection%20Enterprise/WoodWing_Smart_Connection_Enterprise_brochure_EN.pdf.

WoodWing Software. “Woodwing web shop.” http://woodwing.com/en/WoodWing_Store.

Wordpress. “Installing Wordpress.” http://codex.wordpress.org/Installing_WordPress#Famous_5-Minute_Install.