

GRANDON GILL

SECURING THE MUMA JOURNALS¹

When my personal website gets hacked, it is annoying—but I can also view it as a learning experience. But what level of risk am I willing to take when my school's name is on the line?

Grandon Gill, Professor and Academic Director of the Doctor of Business Administration program at the *Muma College of Business* (MCOB) of the *University of South Florida*, pondered the question. A few months before, he had volunteered to take on the creation of two open access journals, to be called the *Muma Case Review* (MCR) and the *Muma Business Review* (MBR). The first of these would publish business discussion case studies and technical notes intended for educational purposes and would exist entirely online. The second would publish research of interest to business practice and would initially publish online and would also provide printed volumes.

Just a week before, Gill had committed to launching the MCR within two months. At the time, this deadline did not seem unreasonable. He already had sufficient case studies to provide a year's worth of content. What he did not have, however, was a specific plan for delivering the content. Originally, he had planned to use the publishing component of the review system that would be handling submission and review of manuscripts. The MCOB's dean, however, had indicated that he wanted a level of branding that could not be accommodated by the review system. Thus, a new solution to the reader-facing front end of the two journals was required.

Gill's instinctive reaction had been to propose a front-end solution that employed *WordPress*TM, a content management system that provided flexible functionality that could be adapted to a journal front end. According to the web survey group *W3Techs*, as of early 2015, nearly a quarter of the top 10 million websites in the world used WordPress. Gill, himself, had used it for years for his personal blog and was comfortable with its capabilities.

As he thought about it further, however, Gill also recalled that his personal website had been taken down at least five times by anonymous hackers. In each case, WordPress had been identified as a likely source of the problem—although that had never actually been confirmed. His dean would not be pleased if similar disruption of the MCOB's flagship journal websites took place. Should Gill be considering alternative solutions? And, even if he did choose WordPress or a similar content management system, there were many possible ways it could be deployed: as a service provider, as an application hosted in the cloud or on a local server. Each had its own pros and cons from a security standpoint. And the costs could be very different. This decision was proving to be far less straightforward than he had first thought.

¹ Copyright © 2016, *T. Grandon Gill*. This case has been reprinted from the *Muma Case Review*, Volume 1, Number 1 and was prepared for the purpose of class discussion, and not to illustrate the effective or ineffective handling of an administrative situation. Names and some information have been disguised. This case is published under a Creative Commons BY-NC license. Permission is granted to copy and distribute this case for non-commercial purposes, in both printed and electronic formats.

Open Access Journals²

Upon their launch, both the MCR and the MBR would become participants in a large and fast-growing academic publishing industry. That industry was built around on the sharing of knowledge, particularly knowledge acquired through rigorous research, validated by peer review prior to publication.

As academic publishing moved away from print towards digital publication during the period from the 1980s to the time of the case, a new type of outlet emerged: open access journals (OAJ). The common principal governing these journals was that the research product should be available for distribution to readers at no cost. Other details, such as who held the copyright of published work, how submissions were reviewed, and how the content was maintained online, varied considerably from journal to journal.

Impetus for OAJ

The emergence of open access journals was driven by a number of trends, all of which tended to reduce the barriers to entry for a new journal:

1. *Cost reductions*: A dramatic decline in the costs of producing digital publications when contrasted with their earlier paper counterparts. These reductions came from the elimination of manual typesetting and artwork, the ability to streamline and automate peer review workflows with online systems that eliminated copying and mailing over course of multiple review cycles, and substantial reductions in printing costs through the use of on demand printers—such as Amazon.com’s *CreateSpace*. Indeed, many open access journals chose to remain entirely digital.
2. *Improvements in search technology*. Prior to the internet, researchers were first obligated to use paper-based card catalogs and, subsequently, library-owned clunky database systems to search for relevant research literature. With the advent of high power search engines, such as Google Scholar, it was feasible to find articles and books nearly anywhere on the Internet.
3. *A growth in global higher education*. According to a Unesco (Altbach, Reisberg & Rumbley, 2009) report, during the 7 year period between 2000 and 2007, global higher education enrollments increased by 53%, a trend that did not appear to be abating. Concurrent with that growth, increasing need for research faculty was experienced. Because these faculty were normally expected to publish their scholarly work, a growing need for research outlets was experienced. In fact, as early as 1950 there was an estimated 60,000 journals and, growing at over 3% per year, by 2010 that number was likely to be closer to 1 million, although the exact number was unclear (Larsen & von Ins, 2010, p. 576). The OAJ segment contributed to this explosion of research content.

OAJ Business Models

Even with dramatic cost reductions, the lack of subscription revenues made it challenging to monetize an OAJ. Three models, individually and in combination, tended to be used:

- 1) *All volunteer model*. Authors, editors and reviewers all contributed their efforts at no cost. Typically, the unavoidable costs of running the journal—such as website hosting fees and review system costs—were provided by an institution or through donations.

² This section is adapted from “A Note of Scholarly Open Access Publishing” (Gill, 2016).

- 2) *Submission fee model.* Authors are charged a fee for submitting a manuscript, whether or not it is accepted. These may vary by submitter category, for example some journals charge more for submissions from the U.S. than they do for submissions from developing countries.
- 3) *Publication fee model:* Authors are charged a fee in order to publish an accepted manuscript. These fees may be a flat rate, charged on a per-page basis, include surcharges for types of content (such as color graphics) and may also include an extra fee for making an article open access (e.g., *Science*, published by AAAS, charges a \$3000 base fee or \$4000 if the article is to be published open access).

New models were continually being developed. For example, *ScienceInsider* reported that a new journal, *Collabra*, was being launched that planned to pay peer reviewers with the goal of helping them afford publications fees when they submitted (Chawla, 2015). Some journals also employed advertising on their websites.

It should also be noted that some of these fee models were also employed by journals, some of which are quite prestigious, that are not open access (*Science* being an example). What distinguishes the open access journal is the fact that it had few other sources of revenue.

Predatory Publications

Because of the potential value of publications to an academic career, many faculty members feel a pressing need to publish. As the cost of setting up a scholarly publication outlet declined, it became possible to establish a profitable scholarly journal that subsisted entirely on submission and/or publication fees. The drawback of such an approach was that it created a strong incentive to accept nearly any submission, regardless of intrinsic scholarly merit. In many cases, blatant abuses—such as plagiarism—were tolerated in order to accumulate fees. As such journals started to emerge, they became known as predatory publishers. Any OAJ wishing to retain its credibility needed to take considerable care in ensuring that its practices and policies did not land it on one of the predatory publisher lists. Once included, it could take years for the journal to rebuild its credibility. Ensuring that the journal's website was adequately protected against security threats would clearly be an element of these practices.

The Muma Journals

The decision to launch two new journals at the *Muma College of Business* (MCOB) was made in late spring 2015. Both journals were to be open access, using the all-volunteer business model, with additional required funding coming from the new *Doctor of Business Administration* (DBA) program and from the MCOB itself. The impetus for these journals sprung from both curricular need and a sense of opportunity.

Background

The idea of launching the MCR and MBR journals occurred during the first semester of the new *Doctor of Business Administration* (DBA) program at the MCOB. The DBA program was a three year doctoral degree targeting working professionals with at least 12 years of experience, at least 5 of which were at the executive or senior management level. Participants went through the program as a cohort and, during their first two years, most of their activity involved a taking a variety of courses focused on research methods and on the strategic research areas of the MCOB (e.g., analytics, creativity and innovation).

As part of the first two years of the program, participants took three “publication courses”, as shown in Exhibit 1. In these courses, each student was required to author a manuscript suitable for submission to a journal. Ideally, they would also submit it. During the first of these courses, in spring 2015, they were

asked to author a discussion case study—ideally one that related to their job. This type of case study had a number of characteristics that distinguished it from other types of case studies. Specifically, it was built around describing the context of a decision that needed to be made. Unlike a "best practice" case—describing the benefits of a process or technology, such as often could be found on company websites—or an "incident" case—which typically analyzed a situation where things went bad—a discussion case did not tell a complete story. In fact, it differed from example cases in three important ways:

- 1) It was open-ended (i.e., it did not describe the decision that was actually made or the consequences of that decision)
- 2) It rarely had a "right" answer--since most interesting decisions did not, although they often had a lot of bad possible choices
- 3) It was specifically designed for the purposes of stimulating discussion in the classroom—which could be seen as the justification for (1) & (2).

The first step in the publication course process involved creating the first page of the case, which summarized the decision that was the focus of the case (the current case serves as an example). These were collected, copied and distributed to all 25 participants. Moez Limayem, the dean of the MCOB, was also given a copy. His immediate reaction was to propose launching a journal which could be used to publish the cases and also serve as a broader outlet for other DBA publication courses and for researchers from outside the cohort.

Gill immediately agreed to the concept, since he knew of very few scholarly outlets that published discussion cases. While these cases were used in many business programs, they tended to be published through outlets such as *Harvard Business School* (HBS) publishing, which charged prices ranging from \$4 to \$14 case, depending on the specific case and the type of customer (students paid the lower rate). In light of that, in 2011 he had launched the open access *Journal of IT Education: Discussion Cases*, which published about a dozen cases a year. Unfortunately, only a small fraction of the DBA cases were in the information technology area, so it was not a suitable outlet for most of the students. Thus, the *Muma Case Review* was conceived.

As Gill and Limayem spoke further, it became clear that the MCR would only address part of the publication problem. The logical audience for discussion case publications would be faculty members interested in using the cases for classroom purposes. They were not well suited for casual reading, nor would they be of much interest to practicing managers unless the organization being profiled was a competitor or potential competitor. That meant that the two remaining publication courses, both intended to produce student manuscripts describing research that would be of interest to practice, would not be a good fit with the MBR. Unfortunately, neither Gill nor Limayem could think of plausible existing outlets that focused on delivering research to the practitioner audience. While such outlets, such as the *Harvard Business Review* and *Sloan Management Review*, did exist, they were typically not receptive to submissions from students—even students with substantial executive experience. Thus, the need for a second journal, the *Muma Business Review*, was identified.

Implementation

Part of Gill's willingness to organize the launch of the MCR and MBR was his past experience with OAJs. This experience had been acquired through his involvement with the *Informing Science Institute* (ISI), where he served as a Governor and Editor-in-Chief of two open access journals. Having gone through the process of both editing and launching an OAJ, he realized that what they planned was plausible—provided they had expert guidance (such as that the ISI could provide).

In order to take advantage of ISI's expertise, Gill proposed using the publication review system that had been developed for ISI by one of its members, originally named Icarus. The system, which he had used for several years, provided very nice functionality for managing the flow of papers from submission to publication, including all aspects of the peer review process. It allowed registered users to take on multiple roles (e.g., member, author, reviewer, editor) and provided a dashboard and user-friendly interface for managing workflow (see Exhibit 3).

The ISI review site had recently been revised to permit use by outside journals. The interface, however, lacked the level of built-in customizability that would allow the MCR and MBR to fully control the reader's experience. While the review system's developer felt that such customization features could be incorporated into the site's design, it would likely take a considerable amount of time before that component of the system could be developed and tested. For that reason, Gill proposed that a three element architecture be employed.

Architecture

The three elements of the architecture are illustrated in Exhibit 4. Moving from bottom to top:

- *The review system:* The ISI review system would be used to manage the submission, review and production process. It would be customized to identify the Muma journals and would maintain a separate pool of members, reviewers and authors from the ISI system. Otherwise it would keep existing functionality excepting the fact that it would not publish automatically to the server holding ISI's existing journals.
- *The publication repository:* This system would hold the actual .PDF publications of the two journals. It would maintain a permanent web address (which could be pointed to different servers as needed) and would require minimal functionality—support for file transfer protocol so that articles could be uploaded by the publisher and accessed by readers. This site would also hold a file containing metadata—the information needed by sites such as Google Scholar to ensure that articles were properly cataloged and could be found by other researchers.
- *The reader front end:* This site would serve as the public interface to the journal, where readers could search for and access articles. It would also provide links to the review system and to any additional educational content that might be useful for authors and reviewers.

The front end element of the system would be particularly crucial to each journal's success. At a minimum, potential readers needed to be able to:

- Find articles or cases in their area of interest quickly.
- Browse the list of articles to determine if any seemed to be of interest without having to open them up.

In addition, a nice-to-have feature would be the ability to link to social media feeds, such as Facebook, Twitter, LinkedIn and Google+. By pushing content to these environments, additional readership for the article or the journal as a whole could be generated. Similarly, a system that supported reader comments might make the journal site more interactive.

Gill's initial instinct was to build the front end in WordPress, a widely used content management system that he had worked with for a number of years, and used on his personal website. From past experience, he knew that administering the site was not too difficult, and that it was also a widely distributed skill

owing to the popularity of the product. To test this out, he created a mockup of the site, using a collection of his own cases as test data, shown in Exhibit 5. Using the ability to categorize article posts by class, he was able to implement an interface that made it easy to look for cases in a particular domain or on a particular subject—a feature that would be critical to instructors. It would also be easy establish links to social networks, since those features were built in, as well as editor-moderated comments.

On the negative side, the free hosting version that he used to create the mockup did not offer much customizability. This could be remedied, he thought, with one of the paid versions. But, before making any final decision, he felt the need to step back. The one element of the architecture that he had thus far failed to address was the security of the system. He recognized, however, that the MCR and MBR would not survive long if they were frequently hacked, defaced or taken down. Was the solution he was proposing secure and, if not, what could he do about it?

WordPress

WordPress was an example of a content management system (CMS). In broad terms, the role of a CMS was to allow non-programmer users to develop and maintain a website with advanced functionality. Although it was originally developed as a platform for bloggers—individuals that regularly posted web content or opinions on specific topics—over time its functionality had grown to the point where many, if not most, web publishing tasks that did not require access to external systems or databases could be accomplished through the application. By 2015, an estimated 25% of the 10 million most popular websites in the world used WordPress to deliver their content.

WordPress Architecture

WordPress was developed as an open source project, using PHP (a web-based scripting language) that allowed it generate dynamic webpages. The content that was displayed on a WordPress site was stored on a local database—most commonly, the open source MySQL—that could be accessed by PHP files. When a user entered (or clicked) the URL (i.e., http:// address) of a WordPress site in his or her browser:

- The browser sent a request to the web server hosting the site.
- A base PHP file was then loaded by the server, which began to run the PHP script contained in the file.
- Within the PHP script, calls were made to the underlying database and to other PHP files to acquire additional content. As the script ran, an HTML webpage file was generated.
- The newly rendered HTML webpage that was sent back to the user’s browser and displayed.

Much of the popularity of WordPress was a result of its customizability. When a user set up a WordPress site, a base level install established the database and all the scripts necessary to launch a fully functional site—albeit with no content. The user then had the ability to customize the site in a variety of ways. The most popular of these were:

- *Themes*: Collections of PHP files and style sheets that gave the WordPress site a distinct look and feel. Many were also designed to be a good fit with a particular type of site, such as a project management site or a holiday site. Most themes allows some level of user customization, such as the ability to change fonts, backgrounds, logos and active components (such as menus). These were mainly produced by third parties, although the base installation of WordPress provided several. Additional themes could be free or available for purchase.

- *Plugins*: Collections of code that could be added to the site to accomplish specific tasks, such as adding a shopping cart or creating links to connected social networks. Like themes, these were mainly produced by third parties, and could be free or available for purchase.
- *Widgets*: Often provided in conjunction with themes or plugins, there were active components that could be placed in specific regions of a page, normally limited by the theme.
- *Customized code*: Because it was an open source platform, the developer of a WordPress site could modify existing PHP files within a theme, add files, or develop their own theme entirely—essentially crafting their own custom website.

As a general rule, the greater the level of customization, the greater challenge of system maintenance. Themes and plugins tended to be updated frequently. Reasons for such updates included aesthetics, functionality and security. Unfortunately, these updates tended to overwrite developer customizations when old PHP files were replaced. That meant that considerable reworking of past customizations was often necessary in order to bring an updated site back to its pre-update level of performance.

Deploying WordPress

Another factor impacting the customizability of a WordPress site was how it was deployed. Broadly speaking, there were five alternatives:

1. *Free hosted site on wordpress.com*: Under this option, the user was constrained in a number of ways, including 3 GB maximum storage, very limited customizability (e.g., no third party plugins) and the requirement to use the wordpress.com domain (e.g., <http://your-site-name.wordpress.com>).
2. *Paid hosted site on wordpress.com*: Similar to the free site (see Exhibit 6) but providing the ability to alter the style sheets used by a theme—something that could dramatically alter the aesthetics of the site, if not its actual layout—and the ability to point directly to the site (e.g., <http://your-site-name.com>).
3. *Deploy wordpress.org installation on a hosted shared server*: Many web hosting companies allowed a site to install the open source version of WordPress (downloadable from wordpress.org). Because the site had its own database and installation, all forms of customization were supported, including third party themes, plugins and customizations. The cost of this option was very reasonable, similar to that of the wordpress.com hosted option. This was the option that Gill had used on his personal website (<http://grandon.com/blog>). One important requirement that came with this deployment was the regular need to manually install updates to WordPress and all its components. Such maintenance was easy to overlook and just thinking about them caused Gill to go to his blog administration page to run an update (Exhibit 7).
4. *Deploy wordpress.org installation on a separate server*: When hosting WordPress on a shared server, dozens (or even hundreds) of websites could be sharing the same server, leading to both performance and security concerns (to be discussed later). These concerns could be reduced though hosting WordPress on its own server. In the past, this had required a separate machine. More recently, a similar level of isolation could be accomplished through the use of virtual private servers (VPS), a number of which could be installed on the same physical server. Except where server traffic was very high, virtual servers offered performance to dedicated servers at a fraction of the cost. From a site maintenance standpoint, the VPS option was effectively identical to the shared option.

5. *Use a managed WordPress host:* A hybrid of options 2, 3 and 4, some web hosting companies offered managed WordPress sites that took care of many of the regular maintenance activities associated with options (3) and (4). Some hosts supported VPS while others employed shared servers. The cost of these options varied considerably, but were generally several times that of other solutions (starting around \$300/year).

From the reader's perspective, options (2)-(5) were likely to be identical, although substantially more functionality *could* be deployed using options (3)-(5) owing to the availability of third-party plugins.

WordPress Security

The very popularity of WordPress presents security concerns. According to one recent web survey, more than 70% of high traffic WordPress sites had one or more security vulnerabilities (Abela, 2013). In most cases, these vulnerabilities stemmed from a failure to update to the current version of the application. But, by virtue of being open source, WordPress was also vulnerable to what were known as "zero day exploits", where a hacker with access to the underlying WordPress code figures out a way to break it and proceeds to do so prior to the vulnerability becoming known to WordPress developers.

Recent Example

In 2015 unsafe functions calls were made within a script found in one of the WordPress themes that was loaded as part of a standard installation. According to an article on ArsTechnica.com (Goodin, 2015):

Millions of websites running WordPress are at risk of hijacking attacks thanks to a vulnerability that is actively being exploited in the wild and is present in the default installation of the widely used content management system, security researchers warned Wednesday.

The cross-site scripting (XSS) vulnerability resides in genericons, a package that's part of a WordPress theme known as Twenty Fifteen that's installed by default, according to a blog post published Wednesday by security firm Sucuri. The XSS vulnerability is "DOM based," meaning it resides in the document object model that's responsible for how text, images, headers, and links are represented in a browser...DOM-based XSS attacks require the target to click a malicious link, a limitation that greatly lowers their severity. Still, once an administrator takes bait while logged into a vulnerable WordPress installation, the attackers can gain full control of the site. Sucuri researcher David Dede wrote:

What is interesting about this attack is that we detected it in the wild days before disclosure. We got a report about it and some of our clients were also getting reports saying they were vulnerable and pointing to:

[http:// site.com/wp-content/themes/twentyfifteen/genericons/example.html#1](http://site.com/wp-content/themes/twentyfifteen/genericons/example.html#1)

In this proof of concept, the XSS printed a javascript alert, but could be used to execute javascript in your browser and take over the site if you are logged in as admin.

In response to the discovery of the vulnerability, WordPress sent out an urgent update. But, as previously noted, the degree to which sites comply with such requests is limited. This is particularly likely to be true for sites with customization since, as previously noted, modifications to standard WordPress and plugin files tend to be overwritten with each update, meaning that they can require a lot of post-update work.

Common Types of Vulnerabilities

The nature of the WordPress architecture made it prone to a number of different types of vulnerability. These included the following:

- *Reflected cross site scripting (RXSS)*: Users of a site enter text that includes scripting code into an input control (such as a textbox) that assumes that only text has been entered. When the text is rendered on a webpage, the code executes—potentially with malicious consequences. For example, a hacker might message another user with embedded code. If the user receives the message on a browser with Javascript enabled, the script could execute commands that could give the hacker complete or partial control of the unsuspecting user's system.
- *Persistent cross site scripting (PXSS)*: Similar to reflected cross site scripting, except that the malicious script is entered into a control whose content is saved to database, such as the database used to store the posts displayed on a WordPress site. Consequences similar to those of RXSS result each time the content of the textbox is rendered on a web page, such as when a user clicks on a post to read it.
- *SQL injection*: Frequently, user input is used as the basis of a search command that executes on the server database. If not properly validated, it is possible to append additional SQL commands to the input. Most commonly, this allows the hacker to access information from the database for which he or she is not authorized.
- *Buffer overflow*: When text is received from user input, it is generally placed into a holding area in the server's known as a buffer. If more text is provided than the buffer can hold, it can conceivably run over into areas that contain other data (such as scripting code). Through that process, the intended execution of the script can be disrupted or, in the worst case, alternative code inserted by the hacker may be executed.
- *Denial of service*: When hacker code is injected through other means, it can cause the server performance to degrade or to fail entirely. This could be accomplished through overloading the server CPU, exceeding the server's RAM or through writing files to the point that they reach the server's disk capacity limits.

The risk of these and other vulnerabilities could be greatly reduced by two means: 1) careful selection of well-known, highly rated themes and plugins, and 2) by continuously updating the site and its plugins. Even that, however, would not protect against other vulnerabilities present in any virtually system—such as failure to properly protect user passwords. Sites hosted on shared servers were also vulnerable to security failures not of their own making, since it was potentially possible for a hacker to gain control of an entire server through any site hosted on the server.

For the most part, these types of vulnerabilities were greatly reduced for sites hosted on wordpress.com. By drastically limiting what plugins a user could install, what customizations a user could make and by automatically updating all sites to the newest WordPress version, the sources of most threats had been eliminated. Even though these sites were hosted on a shared server, the vulnerability of shared hosting was reduced by virtue of all shared-server sites being similarly secured. While this did not eliminate all possible sources of external vulnerability—a concerted denial of service attack launched by a botnet could potentially bring down any server—it did greatly reduce the intrinsic vulnerability.

Gill could find no reported examples of successful hacks on wordpress.com sites when he searched the web. As a further test, he emailed Jeffrey Beall, the developer of the most well-known list of predatory

journals, hosted on a wordpress.com account (<http://scholarlyoa.com>), asking for his experiences. Gill reasoned that such a site would be a magnet for hackers, since many purveyors of these journals were, effectively, criminals. Beall had replied:

I've been using WordPress since January, 2012 and am happy with it. It optionally uses a two-step authentication process, which I've enabled.

So, for me to log in and edit the page on a new device, I have to enter my name and password, and then the system asks for a number. The number is sent as a text message to my cell phone.

He had reported performance problems on one particular page, however. After examining the page itself—which included several hundred user comment posts that would need to be retrieved from the database each time the page was displayed—Gill surmised that these problems were less likely to be the result of hacking than of being hosted on a shared server.

Gill's Prior Experience

Having run his own grandon.com website for many years, Gill was well aware that security issues were not limited to large, well-travelled sites. On at least four occasions since 2011, when he first began using WordPress, his site had been hacked. None of the attacks seemed particularly sophisticated—three of the four simply replaced the landing page of the site with their own landing page, the fourth recursively copied folders until the site ran out of space. In fact, he suspected that the attacks were an example of what were termed “script kiddies”—simple, generally ineffectual code sent out to millions of websites with the expectation of compromising a small percentage of them.

What Gill could not determine was if WordPress was the source of any of the vulnerabilities that had been exploited in his site. In fact, there were a number of other possible sources, including:

- Attacks launched from other user sites that had been compromised on the shared server.
- Attacks stemming from other applications that Gill had installed on his site, which he used as a testbed for exploring different tools.
- Attacks stemming from vulnerabilities in the WordPress plugins that he had installed and, subsequent to the attack, updated.
- Attacks resulting from some misconfiguration of the file system on his part, entirely unrelated to WordPress.

Unfortunately, hackers typically were not kind enough to document how they took the system down. And, where it was only his personal site that was involved, he viewed the matter as a nuisance, not a catastrophe. He was not sure that the dean would feel the same way if the Muma journal sites came down.

Alternative Architectures

The architecture that Gill had prepared, previously described in Exhibit 4, was far from the only possible choice. In fact, there were a number of possible alternatives for both the editor/reviewer back end and the front end facing the journal's readership. He was less concerned with the intermediate system that would host the actual publication files, as a consequence of its intrinsic simplicity.

Back End Systems

There were a number of alternative approaches that could be taken to managing the editorial process (submissions, reviews, revisions, decisions) of the MBR and MCR. These came in both open source and hosted forms.

Open Source

While Gill was able to find number of abandoned attempts to create online publishing back ends, he could only one of these that appeared to be active: *Open Journal Systems* (OJS), developed by *Simon Frasier University* in Canada. Gill had actually installed this system to test it in 2006, when it was being tested as a back end for the journals being published by the ISI. At the time, the ISI had decided not to proceed with it because its highly structured workflow (see Exhibit 8) would make it nearly impossible to move the existing archived articles to the new site.

From Gill's current perspective, transfer of archives was not an issue. What was of greater concern was the fact that the system combined front end and back end functionality (see Exhibit 8). This would either make it very difficult to customize the reader front end, or would result in publications having two front ends—one created by OJS and one created by Muma. Gill felt this would prove a serious source of confusion for readers.

Gill also had some concerns regarding the potential security vulnerabilities of open source software that was not in widespread use. While a program like WordPress had plenty of eyes looking at new release code, the same could not be said of open source projects with few active users. Prior to working with members to develop its own system back end system, ISI had relied on OpenConf, an open source system for managing both its conferences and journals. Eli Cohen, the ISI founder, related some of the problems he had faced:

ISI had malware problems when we were running OpenConf as the webware for each of our various journal sites. While I was never able to detect exactly how the hackers gained control of our websites, they did bring down the sites a few times over the years and at other times inserted their own malware code into footers. The break-ins could have been the result of insecure coding by OpenConf over its various versions or it could have come from our using a shared server.

Since OpenConf issued new versions to fix security problems, OpenConf possibly was the vector for infection. A more practical problem was that each journal ran its own instance of OpenConf, each with its own database. This meant that data about, say, someone's email addresses was re-entered over and over, as an author and as a reviewer for each journal. Updates made to one place (as reviewer or as author) did not update other places in that journal's database for that person, much less update across all the journals.

When we started, there was no option to host our presence on a virtual private server since they didn't exist. Either we used our own server (or leased space at a datacenter) or use a shared server. We could afford only a shared server. The advantage of a shared server was that a professional was always installing security and performance updates and worrying about connectivity problems. We now rent space on four distinct servers including two VPSs. Our new software uses a unified database to keep track of all people, as colleagues, authors, reviewers, and in their other roles, and processes paper reviewing across all our journals and our conferences. Because it is a purpose-built software designed by us for our particular needs and coded by a professional programmer and since its source code is not available for hacker perusal, we have so far discovered no successful hacking attacks. While a professional hacker might be able to break

in, we have been impervious to kiddy scripts. And because we do not store any sensitive information, we are not as much a target for professional hackers as is a retail establishment.

Hosted Solutions

With tens of thousands of journals managed by commercial publishers, it came as little surprise that a variety of hosted options existed. An example of such a system was *ScholarOne Manuscripts*[™], also known as Manuscript Central, offered by publishing house *ThompsonReuters*. This service was used by many premier journals and therefore had an interface that was familiar to most academic researchers. It had extensive functionality that was customizable to include submission fees, publication fees and multiple rounds of peer review. Although it did not publish a price list, one price estimate posted on the web suggested it began around \$5500 (including a one-time setup fee) for 100 submissions per year, and then increased with submissions. Other hosted commercial solutions appeared to fall into the same general price range.

Slightly less expensive was a hosted version of OJS, ranging in price from \$850-\$2700 per year per journal. While this was more reasonable, the earlier-noted problem of the front-end and back-end being combined remained.

The system being hosted by the ISI would have a similar cost to the hosted OJS, but would build on the existing relationship that had been established between ISI and the MCOB (who had recently hosted its *InSITE 2015* conference in Tampa). Unlike OJS, it would readily allow each journal's front end and back end to be completely decoupled. From a security standpoint, the system was installed on a vastly more secure server architecture than ISI's earlier system and, as a consequence, had thus far not encountered any of the security and spam issues that had plagued its earlier review system.

Front End Systems

With respect to potential front end systems for the Muma journals, the array of choices was much greater. He had initially perceived three basic categories from which he could choose:

1. *Publication front end specifically for academic journals*: There were about half a dozen choices in this area, nearly all of which had been developed by universities or library systems. Owing to their special purpose, these systems tended to be strong in terms of ensuring articles were properly registered with appropriate content indexes and properly archived. They tended to be much weaker in terms of their ability to create a highly customized front end. As previously noted, at least one of these options—the OJS application—spanned the front end and back end functionality. Because of their specialized functionality and low visibility, Gill had not been able to find any reports of these systems being successfully hacked. He did note, however, that most of the updates listed for OJS were principally security patches.
2. *Content management systems*: In the same broad CMS category as WordPress, a large array of potential content management systems existed—literally hundreds of possibilities. These ranged dramatically in capabilities, customizability and price. Nearly any of them could be adapted to a journal front end, and each had its own feature set. Many were open source, meaning they could be further modified as needed—although this would necessarily increase the difficulty of further upgrades. Many of these systems were substantially less complex than WordPress and were also less well known. This could, effectively, make them a less attractive target to script kiddies and other low end mass produced hacker tools.
3. *Custom coded system*: Gill anticipated that the MCR and MBR would each publish fewer than a hundred new articles in any given year. Given such small volumes, it would be quite plausible to

add new articles to a table manually. While this solution would provide few nice-to-have built-in capabilities—such as automatically posting to social networks as new content was added—such systems could limit access to a single user, who would be responsible for keeping it updated. By limiting the site to static webpages—i.e., consisting only of HTML pages that did not access databases, provide forms or utilize other sources of dynamic content, security issues relating to the site’s front end would be minimized. This solution would also allow for the greatest freedom of visual presentation and rudimentary search capability could still be added through use of a Google search tool that limited search to the site itself.

The USF CMS

As Gill had continued to pursue the investigation, a late contender for hosting the journal’s front end was proposed: USF’s own content management system. The MCOB’s Director of Marketing and Communications, Lorie Briggs, had encouraged consideration of this option. She had argued that a key objective of launching the Muma journals was to build the MCOB’s brand. What better way than to host the journals on Muma’s own site?

Gill had familiarity with the school’s CMS based upon both his own use of the system for the DBA program and his past research (e.g., Gill, Long & Walpole, 2012). He had not initially considered it because it tended to be quite restrictive in terms of what could be presented and in terms of who was allowed to access it. This policy served both the needs of consistent branding and reducing the site’s potential vulnerability. The MCOB site administrator, Jason Zipperer, indicated that he knew of no successful attempts to hack the site in the two years since it had been implemented. He also indicated that it might be possible to adapt the existing function set of the CMS to the needs of the journal. Zipperer was investigating this possibility at Gill and Brigg’s request.

The Decision

With nearly 30 already peer reviewed case studies in the queue, waiting to be processed and published, Gill was in a hurry to get the first site launched. He knew, however, that a serious mistake in the choice he made would take a long time to correct in the future. The challenge, he felt, was balancing the needs of security against the key objectives of the two journals, which were to:

1. Provide a quality outlet in which the work of MCOB DBA participants could be showcased.
2. Provide local, national and international visibility for the unique activities of the MCOB.
3. Attract submissions from researchers and DBA programs at other institutions who wanted to publish types of work (i.e., discussion cases, interdisciplinary research articles focused at practice) that were not a good fit with other academic journals.

Over time, he expected that the priorities placed on the three objectives would change. In the near future, the first would dominate. Not only would it benefit the DBA program, but it was also nearly impossible to have a successful journal launch if an initial source of content was unavailable. In the long term, he expected the last priority to dominate; neither he nor the MCOB’s dean wished to see the Muma journals characterized as being little more than a local newsletter.

The challenge presented by the decision was the large number of potentially important criteria, security included, that needed to be assessed for each option. In Gill’s mind, these included the following:

- *Fit with Key Objective 1:* The back end needed to be flexible, since alternative peer review approaches might be employed for some submissions. In particular, the peer review capability

provided by Canvas, the USF learning management system, had been used to refine some of the existing cases. Assuming that this type of bypass of the journal back end editorial system might take place in the future, the system chosen needed to be adaptable.

- *Fit with Key Objective 2:* The system needed to provide a professional, searchable interface. It would also benefit from connectivity to social media.
- *Fit with Key Objective 3:* The system should be encouraging to outside submissions. If the system was too tightly tied to USF, for example, other researchers and programs might not see the value of submitting. Only the most prestigious universities—such as HBS with its *Harvard Business Review*--could use their name to attract outside participation. Even those universities maintained a separate entity to manage their journals.
- *Ability to categorize publications flexibly:* While the MCR and MBR were intentionally interdisciplinary in their focus, most researchers and instructors participating in functional disciplines. Since these would be the principal users of MCR discussion cases, it was critical that cases could be classified by functional area and by topic. These coverage areas and topics would necessarily need to evolve over time.
- *Ease of update:* The process of adding each new published article and case to the front end would need to be simple, so it could be accomplished by editorial staff. Equally important, it was critical that the data associated with articles—such as categories—could be modified over time. For example, when a new topic was added to the site's list, it is possible that it might prove relevant to some cases that had already been published. Simple update procedures were particularly critical in a university environment where transient student assistants often were involved in developing and maintaining web sites. Each time such an assistant left, a new assistant needed to be trained.
- *Ease of search:* The site needed to allow potential readers to find relevant cases and articles quickly. Ideally, the front end should be inspection-usable, since managers (and most academics) were unlikely to wade through detailed advanced search instructions.
- *Aesthetics:* The approach should provide the flexibility needed to create a professional, visually appealing site. It should also be modifiable without extensive technical knowledge, since it was impossible to know how the site's needs and branding might change. For example, a complete rebranding had been required in October 2014, when the USF College of Business became the USF Muma College of Business.
- *Upgradability:* The approach should allow for upgraded capabilities to be added easily in the future. These might include performance upgrades—if the sites began to attract large traffic volumes—and the ability to add new functionality. For example, the MBR might eventually offer paid subscriptions to its print edition and the MCR was expecting to publish collections of related cases in book form. Both of these might require a shopping cart application be added.
- *Low internal vulnerability:* Whatever approach was taken, the vulnerability to internal threats—such as cross-site scripting—needed to be kept to a minimum.
- *Low external vulnerability:* Sources of vulnerability, such as those resulting from compromise of a shared server, should be minimized.

- *Affordability*: As open access journals that neither charged submission fees nor publication fees, the Muma journals would not have any revenue sources beyond the MCOB itself. The costs of running the journal—particularly for printing, mailing, proofing and production—were already substantial. Care needed to be taken to ensure that these costs did not grow excessively as a consequence of their digital presence on the web.

Gill recognized that these criteria necessarily competed with each other. But none could be ignored—particularly security. He could imagine the look on his dean’s face if the journals ended up on the predatory journal list as a result of constant security failures. It would not be pretty. By the same token, however, the same outcome could result if the journals started charging publication and submission fees to cover the costs of high end security and professional web developers. What he needed was a solution than minimized unnecessary risks without compromising the journal’s future.

References

- Abela, R. (2013). Statistics show why wordpress is a popular hacker target, *WordPress Security News*, 24 September, Retrieved from: <http://www.wpwhiteseconomy.com/wordpress-security-news-updates/statistics-70-percent-wordpress-installations-vulnerable/>
- Altbach, P. G., Liz Reisberg, L., & Rumbley, L. E. (2009), Trends in global higher education: Tracking an academic revolution, *UNESCO*, Retrieved from <http://www.uis.unesco.org/Library/Documents/trends-global-higher-education-2009-world-conference-en.pdf>
- Gill, T. G. (2016). A note of scholarly open access publishing. *Muma Case Review*, 0(2), 1-15. Retrieved from <http://pubs.mumacasereview.org/resources/MCR-00-02-NoteOpenAccess-1-15.pdf>
- Gill, T. G., Long, K., & Walpole, D. (2012). USF web content management system, *Journal of IT Education: Discussion Cases*, 2(8), 1-27. Retrieved from <http://www.jite.org/documents/DCVol01/v01-08-USF.pdf>
- Larsen, P. D., & von Ins, M. (2010), The rate of growth in scientific publication and the decline in coverage provided by Science Citation Index. *Scientometrics* 84, 575–603 DOI 10.1007/s11192-010-0202-z

Biography



Grandon Gill is a Professor in the Information Systems and Decision Sciences department at the *University of South Florida*, where he also serves as the Academic Director of the Doctor of Business Administration program at the *Muma College of Business*. He holds a doctorate in Management Information Systems from Harvard Business School, where he also received his M.B.A. His principal research areas are the impacts of complexity on decision-making, the diffusion of academic research findings and applying the case method to STEM education. He is currently Editor-in-Chief of *Informing Science: The International Journal of an Emerging Transdiscipline* and an Editor of the *Journal of IT Education*. He is the founding editor of *Journal of IT Education: Discussion Cases*.

Exhibit 1: USF DBA Curriculum

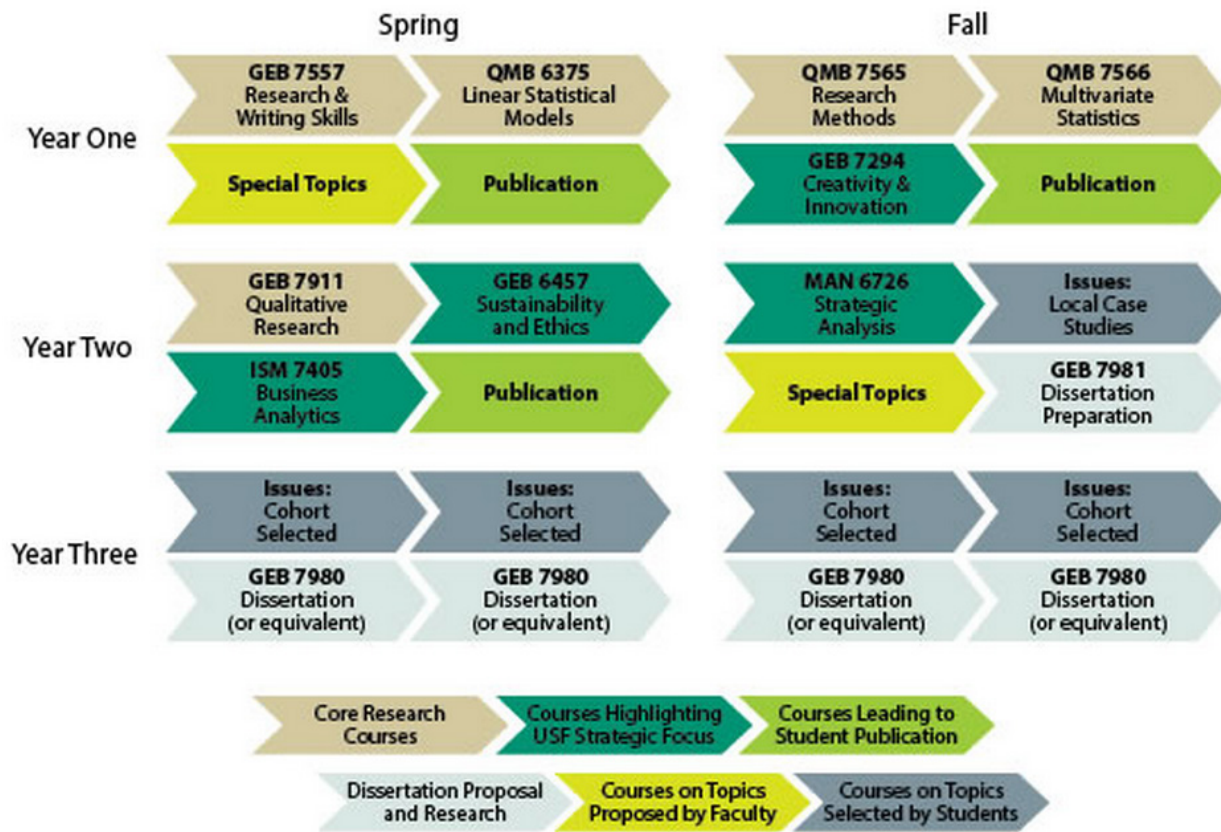


Exhibit 2: Informing Science Institute

INFORMING SCIENCE INSTITUTE
Exploring Better Ways To Inform

Contact Us | FAQ | [LOG IN](#) | [SIGN UP](#)

ABOUT | NEWS | JOURNALS | CONFERENCES | COMMUNITY | SHOP | PUBLICATIONS | Search ...

An international association advancing the multidisciplinary study of informing systems

[LEARN MORE](#)

11 Journals
2108 Publications
2683 Researchers
702 Reviewers

I^SITE 2016 CONFERENCE
Jun 27 - Jul 1 2016, Vilnius Lithuania
[DETAILS](#)

Journals | ISI • Partner • All »

InformingSciJ	JITE:Research	JITE:IIP
JITE:DC	IJELL	IJIKM
IJDS	IISIT	IF

Publications | Featured • Recent • Popular • Browse All »

- Adult Learning and Doctoral Student Research Forum Participation: Insights into the Nature of Professional Participatory Experience**
Joellen Coryell, Kayon Murray-Johnson
IJDS, Volume 9, 2014
- Factors Impacting Teachers' Adoption of Mobile Learning**
Kathryn Mac Callum, Lynn Jeffrey, Kinshuk
JITE:Research, Volume 13, 2014
- Action-Guidance: An Action Research Project for the Application of Informing Science in Educational and Vocational Guidance**
Antonio Cartelli
IISIT, Volume 1, 2004
- Teaching Information Quality in Information Systems Undergraduate Education**
Omar E. M. Khalil, Diane M. Strong, Beverly K. Kahn, Leo L. Pipino
InformingSciJ, Volume 2, 1999

Source: <http://www.informingscience.org/>

Informing Science Institute

Exploring Better Ways To Inform.

About ISI

The Informing Science Institute (ISI), founded in 1998, is a global community of academics shaping the future of informing science. Submitting or publishing papers in any of ISI's peer-reviewed online academic journals is free. With the help of sponsoring institutions, ISI now hosts the highly regarded I^oSITE conference in a variety of international locations twice a year. ISI electronic publications and e-books are available for free. Hard copies of ISI books and publications are available in the [ISI Shop](#), where all proceeds enable ISI to continue sharing knowledge online free of charge.

ISI Research Topics

ISI encourages the sharing of knowledge and collaboration among the wide variety of fields, often using information technology to advance the multidisciplinary study of informing science. These areas can include Business, Communications, Communicating Meaning, Community and Society, Computer Science, Data Management, Distance Education, eCommerce, Education, eLearning, Government, Health Care, History, Information and Library Science, Journalism, Justice and Law, Mathematics, Management, Philosophical Issues, Psychology, Public Policy, Sociology, and Human Resources.

Recommended Read: [A Philosophy of Informing Science](#)

How It Works

Anyone can submit an article to an ISI journal for review, but we require that you become an ISI colleague (free) or ISI Member. By becoming an ISI member, you will not only support ISI by helping to cover administration costs, but will receive several benefits including access to ISI's Member Directory, discounts on ISI products and services, and much more.

[Join ISI](#)

Quick Facts

- Established in 1998
- 9 academic journals
- 2 intl. conferences/year
- 6000+ colleagues and members
- 75+ countries represented
- 600+ organizations
- 5500+ website visits per month

Source: <http://www.informingscience.org/About>

Exhibit 3: ISI Review System Dashboard

INFORMING SCIENCE INSTITUTE
Exploring Better Ways To Inform

ISI Website | Help | Feedback | Log Out

Grandon Gill
ISI Member
PROFILE

- DASHBOARD
- ARTICLE REVIEWS
- REVIEWERS
- YOUR ARTICLES
- NOTIFICATIONS
- FAST-TRACK
- EVALUATION FORMS
- SETTINGS
- CONFERENCES

Article Reviews

2 Pending	0 Delayed	0 Waiting Publication	14 Withdrawn/Rejected
--------------	--------------	--------------------------	--------------------------

Reviewers

18 Pending Requests	254 Available	6 Unavailable	4 In Watchlist
------------------------	------------------	------------------	-------------------

Notifications

NEW	Editors: What is job entails	8 Sep
NEW	Editors-in-Chief: What to do if you upload the wrong PDF	31 Aug
NEW	Your ISI Membership - New Discount Code for 2015/6 Academic Year at ISPress.org	13 Jul
NEW	EIC: How to handle DELAYED development letters	30 Apr
NEW	EIC: Select "Needs Revision" instead of "Accept for Publication" until ...	29 Apr

Exhibit 4: Proposed Architecture

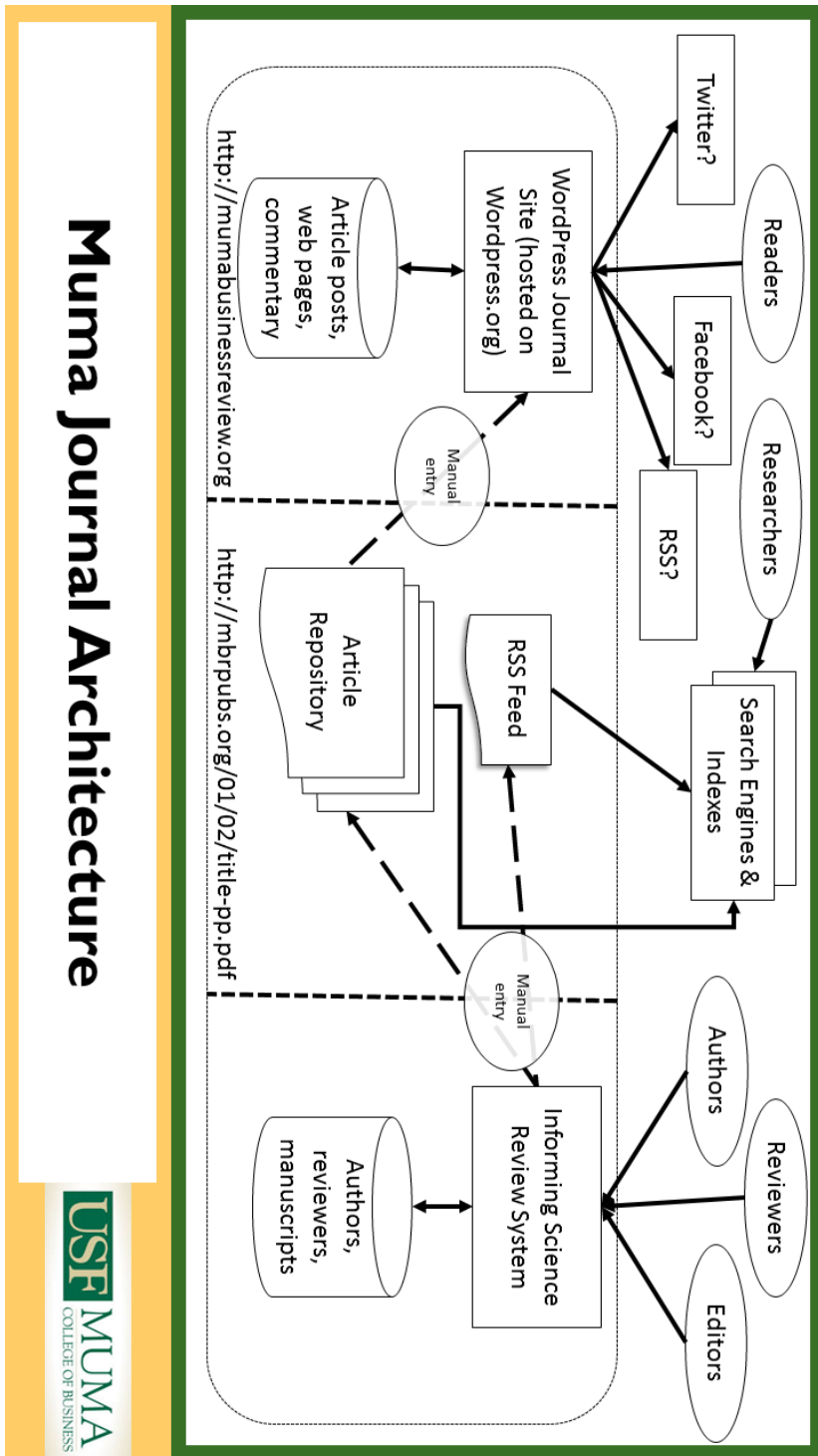




Exhibit 5: MCR Mockup in WordPress


HOME CASES TECHNICAL NOTES ABOUT MCR FOR INSTRUCTORS FOR STUDENTS SUBMIT A CASE





Muma Case Review


CATEGORY ARCHIVES: HIGHER EDUCATION



 **POSTED BY**
GRANDON GILL

 **POSTED ON**
SEPTEMBER 20, 2015

 **POSTED UNDER**
2010, HIGHER
EDUCATION, MIS

 **COMMENTS**
LEAVE A COMMENT


A TALE OF THREE CLASSES: CASE STUDIES IN COURSE COMPLEXITY


A programming course, taught by two different instructors, and a case method capstone course are compared to demonstrate the complex interaction between students, instructors, content and delivery method.


By Gill, T.G. and Jones, J.


<http://grandon.com/publications/GillJones-2010.pdf>


[Edit](#)



 **POSTED BY**
GRANDON GILL

 **POSTED ON**
SEPTEMBER 20, 2015

 **POSTED UNDER**
2013, HIGHER
EDUCATION, MIS

 **COMMENTS**
LEAVE A COMMENT


SYSTEMATICALLY EVALUATING THE EFFECTIVENESS OF AN INFORMATION SYSTEMS CAPSTONE COURSE: IMPLICATIONS FOR PRACTICE


A research paper describing a case method course.


By Gill, T.G. and Ritzhaupt, A.


<http://grandon.com/publications/Gill-Ritzhaupt-2013.pdf>


[Edit](#)



 **POSTED BY**

 **POSTED ON**

 **POSTED UNDER**

 **COMMENTS**

CATEGORIES

Higher Education ▾

COVERAGE AREAS

1997 1998 2006 2010 2012

2013 2014 Agribusiness

Cheating Engineering Financial

Services Healthcare

Higher Education Human

Resources Marketing **MIS**

Online Learning Small

Business Social Media

Startup Strategy

ACCESS

Site Admin

Log out

Entries [RSS](#)

Comments [RSS](#)

WordPress.com

Exhibit 6: WordPress.com Plans

Free	Premium	Business
Great for basic blogging	Great for pro bloggers	Great for businesses
\$0 <i>for life</i>	\$99 <i>per year</i>	\$299 <i>per year</i>
Free	Upgrade Now	Upgrade Now
Free site!	Free site!	Free site!
WordPress.com address	A custom domain	A custom domain
3 GB of space	Advanced customization	Advanced customization
May show ads	Store dozens of videos	50+ premium themes included
Community support	13 GB of space	eCommerce
Learn More	No Ads	Store Unlimited videos
	Direct Email support	Unlimited space
	Learn More	No Ads
		Live Chat support
		Google Analytics
		Learn More

Source: <https://store.wordpress.com/plans/>

Exhibit 7: Updating WordPress

The screenshot displays the WordPress dashboard for 'Grandon Gill's Blog'. The top navigation bar shows the site name, a refresh button, 0 updates, 4 comments, a '+ New' button, and a 'Security' button. The user 'Howdy, grandon' is logged in. The left sidebar contains a 'Dashboard' menu and a list of site management options: Home, Updates, Posts, Media, Links, Pages, Comments (4), Appearance, Plugins, Users, Tools, Settings, Add Link to Facebook, Security, Social Buttons, WPTouch, and Collapse menu.

The main content area is titled 'Update Plugins' and shows a list of failed updates for Facebook links:

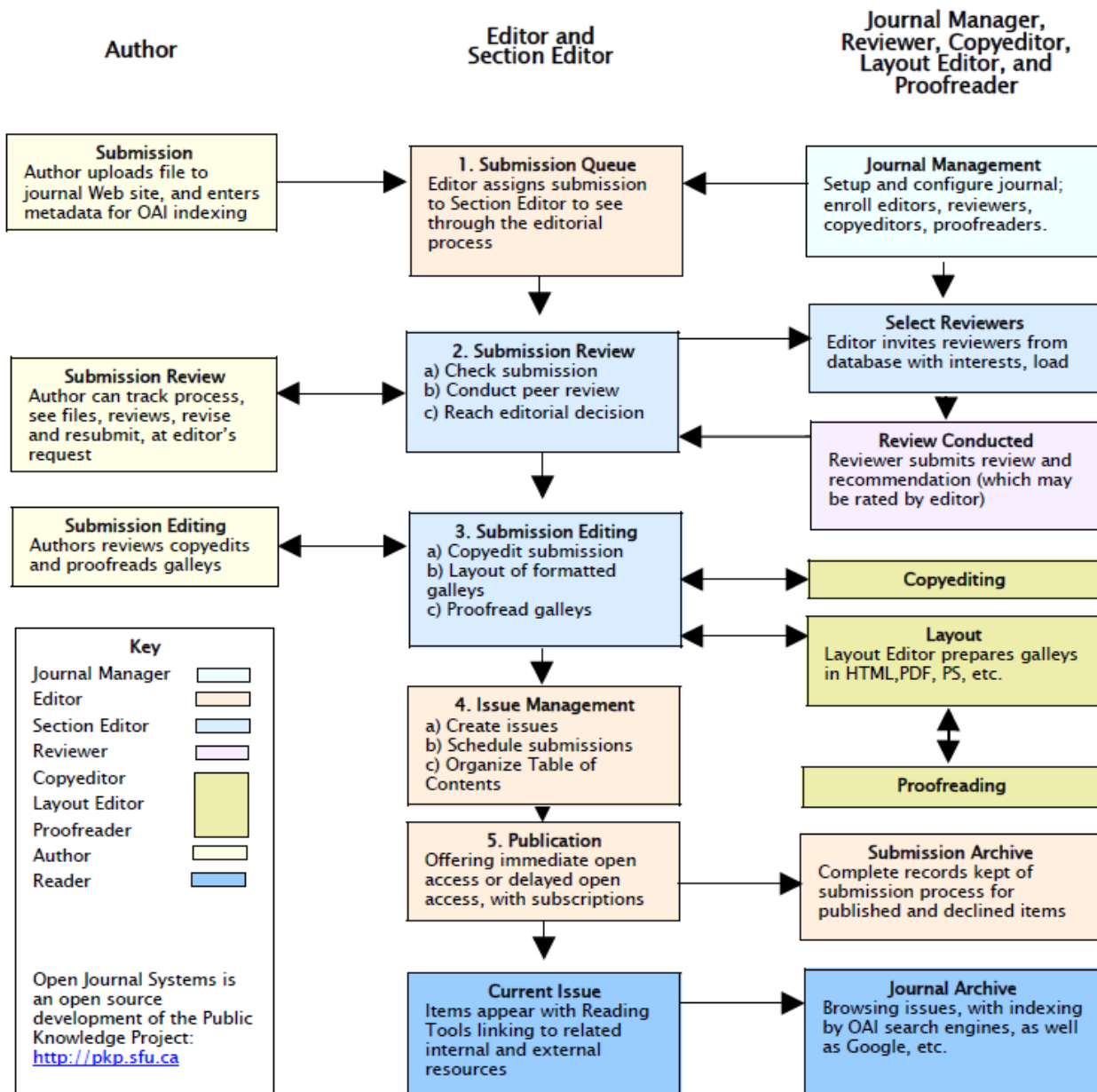
- Add Link to Facebook - [Critical Thinking vs. Decision Making](#): Get me: Facebook error: Error validating access token: The session has been invalidated because the user has changed the password. @ 2013-06-10T11:48:58+00:00
- Add Link to Facebook - [Grandon.com Hacked](#): Get me: Facebook error: Error validating access token: Session has expired at unix time 1341343482. The current unix time is 1360253129. @ 2013-02-07T16:05:29+00:00
- Add Link to Facebook - [Uncertainty and Polarization](#): Get me: Facebook error: Error validating access token: Session has expired at unix time 1341343482. The current unix time is 1349615622. @ 2012-10-07T13:13:42+00:00
- Add Link to Facebook - [The Neighborhood Internet School](#): Get me: Facebook error: Error validating access token: Session has expired at unix time 1341343482. The current unix time is 1344616151. @ 2012-08-10T16:29:11+00:00
- Add Link to Facebook - [My Research: Complexity and "You didn't build that!"](#): Get me: Facebook error: Error validating access token: Session has expired at unix time 1341343482. The current unix time is 1343838756. @ 2012-08-01T16:32:36+00:00

Below the failed updates, the dashboard indicates that the update process is starting and may take a while. It then shows the following successful updates:

- Enabling Maintenance mode...
- Updating Plugin Akismet (1/3)
 - Akismet updated successfully. [Show Details](#)
- Updating Plugin iThemes Security (2/3)
 - iThemes Security updated successfully. [Show Details](#)
- Updating Plugin WPTouch Mobile Plugin (3/3)
 - WPTouch Mobile Plugin updated successfully. [Show Details](#)
- Disabling Maintenance mode...

The process concludes with the message: 'All updates have been completed.' At the bottom, there are links to [Return to Plugins page](#) and [Return to WordPress Updates page](#).

Exhibit 8: Open Journal Systems Workflow



Source: Anonymous (2008), *OJS in an hour: An introduction to Open Journal Systems version 2.2.1.0*, Public Knowledge Project, Simon Frazier University.