Applying Concepts of Bug-Tracking Software to e-
Resource Management in Academic Libraries

Heather Schuster, Jenaya Webb

Heather Schuster is currently completing the Master of Information program at the University of Toronto’s iSchool. Prior to beginning the graduate program, she earned an Honours Bachelor of Science degree from the University of Guelph while working at the Waterloo Public Library and for a Member of Ontario Provincial Parliament. Her long-term interests involve taking the skills of a librarian to non-profit and charitable organizations. Jenaya Webb is Public Services Librarian at the Ontario Institute for Studies in Education (OISE) Library at the University of Toronto. She is a recent graduate of the Master of Information program at the University of Toronto’s iSchool. She also holds a B.A. Hons. and an M.A. in anthropology from the University of Alberta.

Abstract

The addition of electronic resources (e-resources) to academic library collections has added much complexity to traditional workflows. Electronic resource management (ERM) involves intangible items and therefore involves more staff across a broader number of departments as they participate in the processes of a non-linear workflow. Tracking e-resources through the workflow proves to be a challenge. Drawing from a case study on ERM at the Cornell University Library, we suggest that the application of bug-tracking software in the library context has great potential as an ERM alternative. The concepts of the bug-tracking model - status, responsibility, and documentation – provide the environment and functionality needed to ensure good organization and effective communication amongst collaborating staff. Additionally, it allows opportunity for users to engage with e-resource problem monitoring and reporting.

Introduction

Traditionally, the acquisition and management of library resources in academic libraries has been a relatively straightforward processes (Rupp & Mobley, 2007). Today, however, as academic library collections expand to include increasing numbers of electronic resources (e-resources), electronic resource management (ERM) has become a major concern for library staff (Dunham & Davis, 2009). The processes involved in managing e-resources have been characterized as complex and dynamic ones that challenge traditional library workflows and divisions of labour (Collins, 2009; Rupp & Mobley, 2007). Indeed, the processes now required to select, acquire, process, and provide access to online resources have “broken down traditional department boundaries and involve more people from many different library departments” (Collins, 2009, p. 262).

The resulting communication and workflow challenges have resulted in the emergence of a variety of commercial, in-house, and custom-designed ERM solutions (Collins, 2009; Dunham & Davis, 2009; Murray, 2008). In their short case study, Rupp
and Mobley (2007) propose a creative ERM solution: they explain how MantisBT, a free bug-tracking system, is being used as an effective tool for tracking the acquisition and management of e-resources at the Cornell University Library. In other words, they describe how e-resources can be tracked like bugs. We wondered, with so many ERM systems on the market, what makes bug-tracking software a particularly effective tool for ERM?

In this paper, we investigate the usefulness of bug-tracking software for ERM in the academic library context. Specifically, we propose that bug-tracking software can be used successfully in the ERM context because its key underlying processes—tracking the status of an issue, identifying who is responsible for the issue, and collocating relevant documentation—mirror what is needed for effective ERM. We expand on Rupp and Mobley’s (2007) Cornell University Library case study and carefully examine why concepts of bug-tracking software make it useful for the management of e-resource workflows in academic library environments generally. In the first section, 'Challenges of Managing E-resources in the Library,' we introduce the challenges and complexities of managing e-resources in the academic library. In the 'About Bug Tracking' section, we examine the key components and fundamental principles of bug tracking in practice. In our discussion section, 'Applying the Bug-tracking Model in the Library Context,' we take up the Cornell case study as an example, and examine why and how the principles of bug-tracking software can be applied to the academic library context more broadly, and propose bug-tracking software as a cost effective and efficient solution for ERM. Finally, we conclude by suggesting that libraries can further benefit by involving library users in the ERM process. We see this as a key area of further investigation.

**Challenges of Managing E-resources in the Library**

Workflows for managing traditional print resources are more simple than e-resource management and are essentially linear (Collins, 2009; Rupp & Mobley, 2007). However, the characteristics of e-resources differ from those of traditional print resources. It is these differences that lie at the root of the new challenge faced by academic libraries in managing their electronic collections. For instance, e-resources are intangible items: they cannot physically be placed on the desk of a staff member with a routing slip to signify that the item is their responsibility and that some amount of work needs to be done with it (Rupp & Mobley, 2007). E-resources also require non-linear workflows (Collins, 2009; Rupp & Mobley, 2007). Take, for instance, the process of negotiating license agreements. Prior to the acquisition of the e-resource, a two-way dialogue must take place between the library and the vendor until a mutually satisfactory arrangement is found (Rupp & Mobley, 2007). Often, even the price of e-resources needs to be negotiated. Rupp & Mobley (2007) have explained that vendors often base pricing on the projected number of individual users who will access the resource, and their anticipated location of access (i.e. in-library or remote), when setting a price. In addition, e-resources may be bundled in packages so duplication is an issue for academic libraries to consider. Vendors often offer collections of e-
resources in different bundles. For example, Oxford University Press offers a variety of e-resource reference collections including The Premium Collection, which contains different e-resources than their more limited Core Collection. These resources may also be purchased through library consortia. This level of complex negotiation did not exist when resources were only in print.

E-resource workflows tend to involve more staff, found across a broader base of departments, than those of traditional print resources (Collins, 2009). For instance, when we consider reference staff engaging with users and coaching them in the use of e-resources, or technical services or systems staff handling these resources behind the scenes, or librarians involved in collections development, almost all library professionals have a hand in ERM (Collins, 2009). Collins (2009) expresses this well by saying that “in today’s online environment, every librarian is an e-resources librarian” (p. 264). Additionally, the involvement of vendors and library consortia in e-resource workflows demonstrates that staff members are now required to engage with parties external to the library itself more than ever before. Thus, the activities of a greater number of dispersed people need to be coordinated.

Combined, these changes in e-resource characteristics add up to challenges in tracking e-resources throughout the workflow. Collins (2009) explains that e-resource workflows need to be re-conceptualized to match the increasing complexity of ERM in the academic library; communication channels need to be established since they are the key to building effective workflows. To support this, suitable management systems need to be put in place so that the library is not dependent on unmanageable volumes of email and the memory of its staff (Collins, 2009).

**About Bug Tracking**

In the domain of software development, issue tracking, or more specifically bug tracking, has long been an important consideration (Knuth, 1984). Bug-tracking applications are systems designed to track the status and discussion about specific issues or problems (called "bugs") that arise during the development of software code. In our introduction, we asked whether bug-tracking software could be an effective tool for ERM. Here, we examine some of the general concepts of bug-tracking systems in order to better understand why bug-tracking software can be effective in the library context. This section is not a technical overview of all the components of bug tracking software. Rather, it serves, as a survey of some of the key concepts of bug-tracking systems that will later be applied to the library context.

**Framing the Bug-Tracking Process**

The non-technical literature on bug tracking seems to focus on the communicative, collaborative and organizational roles of its processes. Writing about open source software projects in particular, Crowston and Howison (2005) examine bug-tracking systems in order to gain insight into the communications practices of open source teams. Raymond (2001) points to the collaborative and interactive nature
of bug-tracking processes by describing how the people who find bugs can be different from those who understand bugs and those who fix them. For his part, Crowston (1997) anticipates our argument by suggesting that software bug-fixing systems provide a "microcosm of coordination problems and solutions" which can be examined and extended as a potential alternative to coordination problems in other domains (p. 157). Attention to the communicative, collaborative, and organizational potential of bug-tracking systems can help us frame the application of bug-tracking software as a tool for ERM.

**Key Concepts of Bug-Tracking Systems**

According to bug-tracking software manuals, such as *The Bugzilla Guide* (2010) and *Mantis Bug Tracker Administration Guide* (2008), the primary role of a bug-tracking system is to provide a centralized overview of the lifecycle of a bug. Within the lifecycle framework, a bug can be moved from stage to stage (for example, from being ‘new’ to ‘assigned’ to ‘resolved’), be assigned to one or more programmers, and carry with it various types of documentation (such as user comments, tags or keywords). Based on this process, we identify three key underlying concepts for discussion: (i) 'status', which stage the bug occupies in the lifecycle; (ii) 'responsibility', the assignment of the bug to one or more responsible programmers; and (iii) 'documentation', the appending of relevant notes, keywords, or other documentation to facilitate communication about the particular bug. These three concepts in the bug-tracking model form the basis of our discussion of bug-tracking software as a tool for ERM.

A bug’s status is the description of its location in the lifecycle stages between the initial reporting and the final resolution. As the very notion of a lifecycle suggests, bug tracking is an interactive process wherein the status of a bug can be changed, revised, and re-opened at any stage in the process. For a depiction of the lifecycle of a bug, see *Figure 5.1 Lifecycle of a Bugzilla Bug* (Bugzilla, 2010). Additionally, specific lifecycle stages can be customized to suit particular issue tracking projects (The Bugzilla Team, 2010; The MantisBT Team, 2008).

Responsibility for a bug may be assigned to one or more programmers throughout the lifecycle (The Bugzilla Team, 2010). Assignment of bugs to particular team members is a process that must be determined by each individual development team. What is important about the assignment process is that it designates a particular individual responsible for a part of the process and allows other team members to track who is responsible for which bugs. By assigning responsibility to individuals in this way, the bug-tracking model makes the roles of participants explicit and establishes direct lines of communication for collaborative work.

A bug’s documentation includes any additional and relevant information that might help facilitate the process of moving the bug through the lifecycle. For example, The Bugzilla Team (2010) lists a variety of types of documentation including contact information for the person responsible for the bug, the URL associated with the bug, short descriptions of the problem caused by the bug, and any keywords, notes, and
tags added by programmers to the bug’s record. Provided this documentation is clear and consistent with the practices of the development team, it can greatly enhance communication among team members about the nature and history of the bug. Tatham (1999) provides some guidelines for writing good bug reports, which are important considerations for clear bug documentation.

Finally, in addition to the three key concepts discussed above, bug-tracking systems such as MantisBT and Bugzilla include other relevant features such as: search pages that allow users to search for a bug using keywords or other search terms; workflow notifications, where users can set up automatic emails to notify other system users when a bug’s status has changed; and user accounts, which allow the administrators to provide different permissions to various team members (The Bugzilla Team, 2010; The MantisBT Team, 2008). As previously noted, this section is not meant as a technical overview of the functions of specific bug-tracking software applications. However, a quick survey of manuals for bug-tracking applications (such as Chapter 7. Customizing MantisBT of the MantisBT manual) suggests that components like fields, functions, and views can all be customized to suit a particular project. Additionally, bug-tracking software applications like MantisBT and Bugzilla offer a wide range of customizations that can be adopted to suit the needs of the ERM workflow.

Applying Bug-tracking in the Library Context

ERM System Options

Now let’s think back to the challenge of ERM in today’s academic libraries. Since e-resources have added such complexity to collections management, many large libraries have begun to adopt systems designed to assist with ERM (Dunham & Davis, 2009; Jewell et al., 2004). The two main roles of these systems are the centralization (or collocation) of all information relevant to a particular e-resource, and the facilitation of communication between collaborating library staff (Murray, 2008). In the absence of such a system, library staff may rely on email or other de-centralized tools to handle ERM. However, this is inefficient, particularly when staff members are based in different departments and work in a variety of locations (Collins, 2009; Murray, 2008).

Many libraries have adopted commercial ERM systems; however, while these systems may be equipped with vast amounts of functionality, they are not necessarily the most appropriate choice in all cases (Murray, 2008). For small or under-funded libraries for instance, purchasing and maintaining these expensive systems may be unrealistic and unnecessary (Murray, 2008). There are alternatives to commercial ERM systems. For instance, a small college library in South Carolina opted for an open source solution to ERM as a cost-savings alternative (Erb, 2009). Additionally, Murray (2008) has suggested that free Web 2.0 applications such as wikis, blogs, and Google Docs and Spreadsheets, can be pieced together to provide the most basic functionality needed for ERM. Finally, Rupp and Mobley’s (2007) discussion of bug-tracking software as an effective ERM solution provides a touchstone for our discussion.

Bug-Tracking Software for ERM
In the same way that software bugs are said to pass through a lifecycle, we contend that e-resources do as well. In other words, e-resources themselves can be tracked like bugs. *Figure 1* shows our representation of the lifecycle of an e-resource, based on the workflow described by Rupp and Mobley (2007).

*Figure 1: Lifecycle of an e-resource*
Initially, a request is made that an e-resource be added to the library's collection. This request is filtered by library staff and is in turn either rejected or the request is assigned to the staff member who will initiate acquisition of the e-resource. The filtering stage includes initial discussions with potential vendors and the determination pricing and basic information about the resource. License negotiation with the vendor occurs if a license is required. The e-resource is then moved through the various stages of processing in-house, such as activation, cataloguing, testing, and problem negotiation, etc., until it is ready for deployment. At each stage of processing, the e-resource is re-assigned to the staff member responsible for the next stage in the workflow. When processing is complete, the e-resource is made available to library users. Maintenance and evaluation are ongoing.

Regular users of e-resources at an academic library are aware that on occasion problems can arise that cause service interruptions. An example of an error message resulting from service interruption is displayed in Figure 2. There are a variety of reasons for this including, but not limited to, incompatibility issues, problems with finding tools, or service withheld by vendors due to perceived e-resource abuse (Rupp & Mobley, 2007). Figure 1 depicts problems being reported when they are detected after deployment or during routine maintenance and evaluation. When problems are reported, the e-resource is taken back to the filtering stage where it is evaluated once again to assess the reported problem and the value that will be gained by having it fixed. The e-resource may be rejected and subsequently removed from the collection, or it may be re-assigned to the staff member deemed responsible for negotiating the problem.

Bugs and e-resources share some key characteristics. Both are intangible items with non-linear workflows or lifecycles. They are handled by a number of participating individuals who may be working in different locations, and they carry with them information that must be shared.

**Figure 2:**
*An example error message on a library web site (University of Toronto, 2010)*
Previously, we identified three key concepts of the bug-tracking model: status, responsibility, and documentation. When brought to the library context, these concepts create an ideal environment for effective ERM. As was done at the Cornell University Library, e-resources are each assigned a page resembling a bug report. This allows basic information about the e-resource such as title, a brief description, URL, and vendor contact information to be shown. Dialogue concerning the processing of the item, details of the license arrangements, and problems encountered can be collocated there for easy reference. This is undoubtedly more efficient than searching through large volumes of email for the same information. The page can be tagged with keywords to facilitate future searching and reference.

Customized status labels allow all staff to be aware of the exact location of the e-resource in its lifecycle. Even after deployment, the e-resource can be re-assigned to a staff member should a problem interrupt its use. There is never any question of with whom responsibility lies for the next stage in the workflow. This removes the reliance on staff memory that libraries can fall into in the absence of a suitable system for ERM. This reliance on memory is something that needs to be avoided in order to keep workflow effective and efficient (Collins, 2009). In summary, these concepts of the bug-tracking model, applied in the library context, create the environment needed for ERM to thrive: communication is facilitated between staff, and documentation is collocated and accessible. The librarians at Cornell have found that “Mantis has enabled [them] to effectively communicate with one another about electronic resources and keep abreast of the place of any particular resource in the electronic resource workflow” (Rupp & Mobley, 2007, p. 22).

**Involving Library Users? An area for further consideration**

Thus far, we have suggested that the bug-tracking model can provide an efficient and cost-effective solution to ERM. However, we see many areas for further exploration. We conclude this paper by suggesting that in addition to applying the concepts of bug-tracking systems to ERM, libraries might also consider lessons from the open source community in engaging library users in the process.

In the literature on open source development, it has been argued that a particular strength of the process is its ability to respond quickly and effectively to bugs (Crowston and Scozzi, 2008; Raymond, 2001). Paraphrasing Linus Torvalds, Raymond (2001) has famously noted that "given enough eyeballs, all bugs are shallow" (Abstract section, para. 1). This leads us to imagine two ways in which library users could be engaged. When confronted with an e-resource service interruption, library users could be encouraged to report the problem to library staff. For instance, a 'Report a Problem' link could be available at the point of access for all e-resources, taking advantage of the bug-tracking system’s reporting functionality. This link would direct library users to a reporting form, customized by library staff, to elicit key information about the problem and ensure the production of what Tatham (1999) would consider good bug reporting. Alternately, if library staff has already
detected a problem, library users could have the option of clicking on a link associated with the error message (see Figure 2 for a sample error message) to view the status of the issue in question and the expected date of resolution.

Certainly, the extension of ERM to include library users has implications for library policy, staff training, and user education. However, it also offers library managers and staff the opportunity to make the ERM process more transparent to library users. Perhaps most importantly, it offers library managers a chance to realize the benefit of engaging users in ERM. Further investigation in this area could both clarify the needs of users and establish specific ways to engage them in the ERM process.

Conclusion

The challenges of e-resource management (ERM) in academic libraries require solutions that can accommodate the dynamic, distributed character of e-resource workflows. In this paper, we examined the lifecycle of both e-resources and software bugs, and observed that they mirror each other in some important ways. Based on our observations and on Rupp and Mobley’s (2007) Cornell University Library case study, we argue that the key concepts of the bug-tracking model - status, responsibility, and documentation - make bug-tracking software useful as an alternative for the management of ERM workflows in academic library environments. Finally, we suggest that adoption of bug tracking software would allow for effective engagement of the library user community in ERM.

References


