Journal of Business and Technical Communication

http://jbt.sagepub.com/

Integrating Social Media Into Existing Work Environments : The Case of Delicious

Karl Stolley

Journal of Business and Technical Communication 2009 23: 350 originally published online 17 March 2009
DOI: 10.1177/1050651909333260

The online version of this article can be found at:
http://jbt.sagepub.com/content/23/3/350

Published by:
SAGE
http://www.sagepublications.com

Journal of Business and Technical Communication
Volume 23 Number 3
July 2009 350-371

10.1177/1050651909333260
http://jbt.sagepub.com
hosted at
http://online.sagepub.com

Integrating Social Media Into Existing Work Environments

The Case of Delicious

Karl Stolley
Illinois Institute of Technology

This article offers an example case of technical communicators integrating the social bookmarking site Delicious into existing work environments. Using activity theory to present conceptual foundations and concrete steps for integrating the functionalities of social media, the article builds on research within technical communication that argues for professional communicators to participate more fully in the design of communication systems and software. By examining the use of add-ons and tools created for Delicious, and the customized use of Rich Site Syndication (RSS) feeds that the site publishes, the author argues for addressing the context-sensitive needs of project teams by integrating the functionality of social media applications generally and repurposing their user-generated data.

Keywords: *social media; Web applications; RSS feeds; distributed work*

In this article, I describe concrete, theory-grounded steps that technical communicators might take to integrate social media applications (SMAs) into work environments in academic or industrial settings. SMA integration can occur not only by using SMAs like Delicious or Flickr directly at their respective URLs on the Web but by building the functionality and user-submitted content of SMAs into work environments such as Web browsers or project Web sites. In certain limited cases, such as the Delicious add-ons for Firefox that I discuss, SMA integration is as simple as a one- or two-click installation of a small program. But more complex integration of SMA content for addressing the needs of multiple team members

Author's Note: I would like to thank Clay Spinuzzi for his thoughtful comments and guidance on this article. Address correspondence to Karl Stolley, 218 Siegel Hall, Illinois Institute of Technology, Chicago, IL 60616; e-mail: kstolley@iit.edu.

ultimately requires careful and nuanced customization, a "luxury" (Amidon & Blythe, 2008) that workers often lack, even though Hart-Davidson (2002) suggested that technical communicators "should play important roles as researchers, analysts, designers and training specialists in the development of internal communications systems and policies" (p. 466). SMAs therefore present a significant set of customization and design challenges in what are already complexly mediated work environments (Slattery, 2005, 2007).

The social bookmarking site Delicious is a simple, representative SMA for conceptualizing and demonstrating why and how technical communicators would customize the integration of SMAs into other work environments. The site is simple because Delicious houses a limited, specific type of data (bookmarks), and it is representative because the "Web services" (Zeldman, 2007) that Delicious offers in support of integration, particularly Rich Site Syndication (RSS) feeds and a public Application Programming Interface (API), are typically offered by other SMAs, including competing social bookmarking sites such as Ma.gnolia, the popular social-networking site Facebook, and the photo-sharing application Flickr. The customization and integration of Delicious that I present here, then, can also be applied to other SMAs, which collectively provide a model of how technical communicators may "subvert" and "open up" a centralized system "and find ways to build in support for activities that it excludes" (Spinuzzi, 2003, p. 204).

After giving a brief overview of the Delicious Web site's functionality as of this writing, I present a hypothetical case in which sharing bookmarks might be useful to a team of technical writers and other stakeholders charged with revising a set of documents supporting a digital audio recorder. I then highlight the problems posed by the case in light of scholarship that reflects Hart-Davidson's (2002) call for technical communicators' active participation in the design of information technology (IT), which often emerges as "constellations" (Slattery, 2005) of documents and software. To begin thinking about how users can integrate Delicious, I first look at how the open-source Web browser Firefox's add-ons allow individual users to integrate Delicious into their Web browsing activity. I then describe single-user add-ons using Kaptelinin and Nardi's (2006) treatment of activity theory in relation to interaction design, gleaning insights from activity theory as to why and how technical communicators would want to pursue custom uses and integrations of SMAs. That discussion then extends to include an approach based on activity theory and limited code examples for team-oriented integrations of Delicious, in a manner consistent with Spinuzzi's (2003, see chap. 6) ideas for the "design of open systems."

I argue that the rhetorical and technological literacies required to build these extensions are a foretaste of a future in which technical communicators engage in ever-more sophisticated customizations and mashups to integrate SMAs and related technologies into different projects and work environments.

What Is Delicious? An Overview

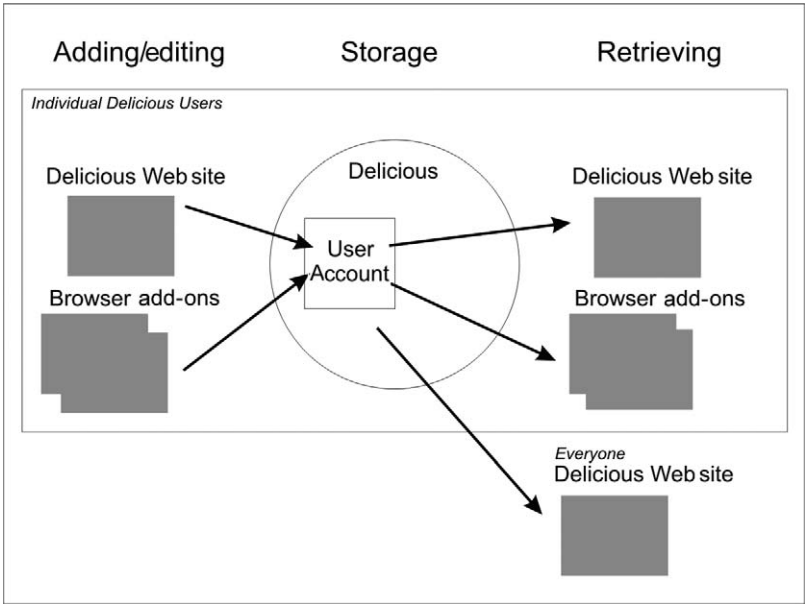
Delicious is an SMA that mediates the bookmarking of Web-available sites and pages and that provides apparently unlimited storage for a user's bookmarks. A bookmark posted to a user's Delicious account consists of the URL of a Web-available resource; a title for the resource (often the bookmarked page's title as described in the HTML); a brief comment, description, or analysis of the site as a note; and one or more tags, which serve to organize the bookmarks and take the place of the rigid folder structure common to browser-based bookmarking systems.

After registering with Delicious, users can immediately begin to add bookmarks to their account via the user interface on the Delicious site itself, which consists of a dedicated Post page that contains a form area for entering the URL that a user intends to bookmark and a Save button. But when the Save button is clicked, rather than saving the URL, Delicious presents a second page with additional forms for entering the site's description, optional note, and tags. Only after this subsequent form is filled out does Delicious actually save the bookmark. But add-ons to Web browsers, which I will discuss, can streamline this process.

After users have added bookmarks to their account, the bookmarks are immediately publicly available via a set of Web pages, based on their account username and tags, displaying the bookmarks in reverse chronological order (listing newest bookmarks first). These Web pages can display all of the bookmarks in the account or sort them on a per-tag or multiple-tag basis. Unlike bookmarks stored locally in a user's browser, Delicious bookmarks are available on any computer connected to the Web and are therefore accessible to the account holder on multiple computers, and to anyone who knows or discovers, perhaps through a Google search, the account holder's URL.

Figure 1 provides a rough depiction of how Delicious functions: Individual users post to their own accounts, either through the Delicious Web site itself or through browser add-ons. Although some users might think of Delicious and other SMAs as Web sites only, Delicious is more precisely service software. Although users can retrieve their bookmarks using the

Figure 1
Basic Model of Delicious's Functionality



Delicious Web site itself, they may also do so through certain browser add-ons that synchronize their bookmarks so that they do not need to visit the Delicious Web site in their browser. Thus, like Web browsers that have built-in search boxes for finding, for example, books on Amazon or Web sites through Google, Delicious has multiple technologies for adding and retrieving information. But the use of one technology (e.g., a browser add-on) does not preclude the use of another technology (e.g., the Delicious Web site, which would be useful when browsing the Web on a public computer). Additionally, as Figure 1 illustrates, everyone (not just individual Delicious account holders) may retrieve bookmarks—from across Delicious or from individual accounts—using the Delicious Web site. In the latter portion of this article, I describe how technical communicators can leverage that open access into alternative means for integrating bookmarks from multiple accounts into a centralized work environment.

Delicious as a Mediating Artifact: An Activity Theory Approach

It is not difficult to imagine a situation in which a team of writers and stakeholders within an organization would need to share briefly annotated links to Web sites and pages as part of research activity on a collaborative project (for discussions of enterprise-level use of closed social-bookmarking systems, such as IBM's Dogear, see, e.g., Braly & Froh, 2006; Millen, Feinberg, & Kerr, 2005). Suppose team members are charged with revamping a complete set of documents surrounding a particular product, like a digital audio recorder, for the forthcoming release of its new model. The team is responsible for revising and improving everything from the recorder's marketing and promotional material to its quick-start guide and full instruction manual. Although customer questions and complaints logged internally by support staff will be useful in any meaningful revision, so might be customer reviews of the audio recorder on centralized e-commerce sites such as Amazon.com and CNET, widely distributed reviews found across the blogosphere, or even tutorials on the Web sites of audio enthusiasts or professionals who use the product. Certain reviews might reveal users to be extremely pleased with a feature of the recorder that the original team of writers had not highlighted but which might prove to be worthy of mention in future promotional materials. Other reviews or how-tos might reveal critical setup steps that were neglected or ineffectively explained in a quick-start guide. Similar Web-available promotional and instructional materials created by makers of competing digital audio recorders might also be of some interest to the writing team.

What is difficult to imagine, though, is that technical communicators would automatically welcome into their work flow a new tool such as Delicious that requires, in its out-of-the-box form, visits to a separate Web site to either add or browse and retrieve bookmarks—a functionality that is already built into every major Web browser. Representative studies by Hart-Davidson (2002), Spinuzzi (2003), Slattery (2005, 2007), and Amidon and Blythe (2008) all demonstrate that technical communicators typically work in complexly mediated environments already, with no shortage of tools.

Slattery (2005) has shown that, amidst all of the tools, documents, and sources of information that make up a work environment, the one commodity in short supply is screen space (see also Czerwinski, Horvitz, & Wilhite, 2004; Hutchings, Smith, Meyers, Czerwinksi, & Robertson, 2004). Slattery

observed that "the limitations of the computer screen were a frequent source of difficulty for managing views" of the different tools and documents making up a writer's "constellation," or "regularly occurring coordination of mediating artifacts." These "constellations" often required ad-hoc strategies such as printing documents, simply "because it can be very difficult to scan a document or flip back and forth between multiple documents on-screen" (Slattery, 2005, p. 356)—a process that Slattery has called "textual coordination." Although Delicious or some other Web site–based tool could, of course, be integrated into a work flow of textual coordination such as Slattery described, the problem at hand is to look at ways to integrate tools not merely into work flows but into other extant centralized tools themselves in part to minimize the amount of new screen space that Delicious or other SMAs require. Used at its native URL, Delicious takes up as much screen space as any other Web page. And here, the actual use of Delicious can be analyzed further in light of the activity of the documentation team.

Understanding Delicious as Three Levels of Activity

Kaptelinin and Nardi's (2006) treatment of activity theory and interaction design includes a highly accessible and aptly named chapter, "Activity Theory in a Nutshell" (pp. 29-72; for a book-length treatment of activity theory specifically applied to technical communication and information design, see Spinuzzi, 2003, especially chap. 2). Activity theory, "in a nutshell," provides an object-oriented approach to human activity, with *object* denoting, among other things, "*objectives* that give meaning to what people do" (p. 66; see also their chap. 6, "Objectively Speaking," especially pp. 138-141, for an extended discussion of object). That is, activity theory seeks to understand meaningful, purposeful human activity, both of "individual human beings and the social entities they compose, in their natural everyday life circumstances" (p. 31). Because activity theory addresses both individual and social participants, it is well (although not exclusively) suited to describing both individual and social uses of SMAs: Posting bookmarks to your own Delicious account while browsing the Web is clearly an individual effort whereas retrieving and making use of other people's bookmarks may be both an individual and a social effort.

In addition to conceiving of activity as object oriented and both individual and social, activity theory establishes a flexible, three-tiered hierarchical structure for the analysis of activity: activities, actions, and operations. Purposeful human *activity* constitutes the highest level on the hierarchy and

can be understood in terms of numerous *actions*, "conscious goal-directed processes that must be undertaken to fulfill the object" of activity, with individual goals being composed of lower-level goals (Kaptelinin & Nardi, 2006, p. 67). In contrast to conscious actions, the lowest level of the hierarchy consists of unconscious, or "automatic," *operations*; Kaptelinin and Nardi noted that, unlike actions, "operations do not have their own goals; rather they provide an adjustment of actions to current situations." For example, striking keys on a keyboard is an operation, especially for touch typists; the actions that keystrokes support include forming words and sentences, which in turn support any number of activities, such as writing an instruction manual for a digital recorder. But Kaptelinin and Nardi also noted that a "common source of operations is the automatization of actions, which become routinized and unconscious with practice" (p. 68): Frequently carrying out an action (bookmarking on Delicious, posting photos to Flickr, updating your "status message" on Facebook or Twitter) will, as a matter of sheer repetition, serve to transform the action into an automatic (and thus unobtrusive) operation.

Beyond repetition, operationalizing and otherwise streamlining actions also occur through tool mediation (Kaptelinin & Nardi, 2006) although I prefer the broader term "mediating artifacts" used by Spinuzzi (2003, pp. 38-39). Kaptelinin and Nardi (2006) stated that "activity theory's emphasis on social factors and on the interaction between people and their environments" centralizes the role of tool mediation or mediating artifacts, the use of which "influences the nature of external behavior and also the mental functioning of individuals" (p. 70). Applied to a more concrete situation of work, Spinuzzi (2003) explained, mediating artifacts "help the workers meet their goals by regulating or controlling their own actions" so that these artifacts ultimately "transform the ways workers conceive of their work activity, solve problems, set new goals, and so forth" (p. 38). Put another way, "natural human capabilities" can be extended by mediating artifacts that ultimately "allow the individual to attain goals that could not be attained otherwise" (Kaptelinin & Nardi, 2006, p. 64).

Applying the hierarchy of activity to the team working on documentation for a digital audio recorder reveals an important insight into human interaction with SMA technologies, namely that SMAs provide functionality (actions) in support of a larger activity. In other words, revising the documentation is the object of the team's activity—not bookmarking sites on Delicious. Although, given the flexibility of activity theory's hierarchy, the use of Delicious could be considered an individual activity, SMAs are more constructively viewed when positioned as supporting tool-mediated action

in service to a larger activity, such as the team's documentation revisions. That is, the bookmarks from Delicious may make no appearance in the actual documentation of the recorder; instead, they serve as an "upstream" literate activity (Hart-Davidson, 2002, p. 457) that precedes and supports the activity of revising documents.

Positioning the team's use of Delicious at the action level of the hierarchy—bookmarking useful sites and pages to aid in the broader activity of documentation revision—also confirms the polymotivation of actions that Kaptelinin (1996) extended from the work of Alexei Leontiev, a foundational thinker in activity theory: "Two or more activities can temporarily merge, motivating the same action, if the goal of one action is a prerequisite for reaching the motives of all of the activities simultaneously" (p. 58). Using Delicious, team members might execute a single action (posting a bookmark) with multiple motives: They might save bookmarks to their Delicious account not only to distribute knowledge to other people involved in a specific project but also to preserve their own discoveries as they search through content on the Web. Delicious creator Joshua Schachter has aptly described the site as an "amplification system for your memory of Web sites" (cited in Weinberger, 2007, p. 92).

Schachter's description raises an important point. SMAs hold potential value not only for users in the workplace but for users managing information in a variety of situations. As I have witnessed in my classes, when students are required to use Delicious or some other social bookmarking service, some students will use their account to store all of their bookmarks, both class-related and personal, which reflects the observation of Spinuzzi, Hart-Davidson, and Zachry (2006) that "personal information management, once primarily a concern of managers, has emerged as a life philosophy that makes no distinction between work and home life—a life philosophy that relies on creating, sorting, and discarding texts" (p. 43). SMAs continue a trend begun by productivity software such as word processors and spreadsheets, which are now as commonly found on home computers as they are on office computers. Individuals may use SMAs in support of personal activities just as much as they would for work and school activities.

And although someone could certainly assert a "distinction between work and home life" by maintaining separate Delicious accounts—one for work, one for personal use—many of us could identify interests that participate in the blurring of that distinction: In my case, a bookmark to a site promoting some digital device I aspire to own might later be of use as an example site worth sharing with my Web-design students. More pragmatically, maintaining

separate accounts might needlessly complicate bookmarking actions for users who have a problem trying to remember in which Delicious account they stored a bookmark. An added benefit to the frequent, repetitive use of Delicious in support of multiple activities is that, like almost any software tool, a user's interaction with it becomes essentially transparent over time so that "the user can focus on his work, while the system—the mediating artifact—remains 'invisible'" (Kaptelinin & Nardi, 2006, p. 79).

Supporting Individual Actions and Operations Through Add-Ons

The activity theory hierarchy's two bottom levels—actions and operations—are clearly identifiable in the Delicious add-ons that users of the open-source Web browser Firefox can opt to install (add-ons of varying quality have also been built for other browsers, including Microsoft's Internet Explorer and Apple's Safari, but because Firefox and its add-ons are available in essentially identical forms for Windows and Mac OS X, as well as Linux, I refer exclusively to Firefox here). The two official add-ons available for Firefox were created by the Delicious team itself although plenty of unofficial add-ons exist (e.g., searching for Delicious in the Firefox add-ons returns Patrick Lauke's "del.icio.us post" add-on) that allow users to post to their Delicious accounts but are less memory intensive than the official add-ons. Both of the official add-ons for Firefox—the del.icio.us buttons or the classic add-on and the Del.icio.us Bookmarks add-on— introduce two new buttons into the browser next to the standard buttons (forward, backward, reload, stop, etc.) in the Firefox Navigation Bar plus a contextual del.icio.us menu between the History and Bookmarks menus.

Although the buttons for Delicious are visually integrated with the browser, the classic add-on's functionality is limited to calling up Delicious Web pages in order to post and view your bookmarks. But instead of redirecting a user to Delicious, the classic add-on's Bookmark on Delicious button opens a smaller browser window (the way a bit of JavaScript would) with a Delicious-hosted Web form that is prepopulated with the URL and title of the Web page in the user's active browser window (which is still visible). Users may edit any of the prepopulated form areas, which is important in cases in which a page provides a confusing or nondescript title in its HTML title tag. And with the active window still visible and containing the resource that the user wanted to bookmark in the first place, the add-on

supports the lower-level action of referring to the page to be bookmarked, perhaps to aid in writing a relevant note about the bookmark.

But the more recently created Del.icio.us Bookmarks add-on features a more robust functional integration of Delicious by periodically syncing and storing bookmark data with Firefox, which allows users to post, browse, and search their bookmarks within the browser itself without ever visiting a page on Delicious. The Del.icio.us Bookmarks add-on integrates with familiar actions that users might already have operationalized from using in-browser bookmarking tools. In addition to providing a customizable bookmarking toolbar that is automatically updated with users' most recent Delicious posts, the add-on features a bookmarks pane, not unlike Firefox's native one, divided into two parts: a list of all user tags and a list of bookmarks. The list of bookmarks can be narrowed and refined by clicking on a particular tag, which reveals only the matching bookmarks in the pane below. This would be useful to the team members working on the documentation for the digital audio recorder, who might want to view only their own bookmarks related to that project. Users of the add-on also have access to their bookmark notes by hovering their mouse over an icon that is present whenever the bookmark contains a note.

And in a striking, instructive design choice, the Del.icio.us Bookmarks add-on even accommodates users who may wish to continue their (operationalized) local bookmarking in Firefox. Alongside the native Firefox Add Bookmark dialog box, a smaller box appears that asks users, "Also save this to del.icio.us?" Users can continue their local bookmarking actions and opt to save their bookmarks to Delicious. This design example shows how thoroughly the functionality of Delicious or other SMAs may be integrated into extant work environments. The browser, after all, is where users experience the Web through searching, viewing, and bookmarking pages. Any mediating tool that supports and operationalizes those kinds of common actions will only be successful to the degree that it fits in with its broader mediating environment—the Web browser.

In other words, Delicious provides the core infrastructure for storing bookmarks; it also, through extensions to the Web browser, offers a less obtrusive means for individual Delicious account holders to post, browse, and search for bookmarks. This infrastructure allows individual Delicious users to customize and control their experience through the selection of one add-on over another until they have found an add-on that mediates the actions that best facilitate their work. Traditional ideas of user-centered design tend to pursue a one-size-fits-all interface based on metaphors that are familiar to all users. Delicious and other service-software SMAs allow

many interfaces with varying configurations and capabilities to best suit the needs of any given user.

Designers of these browser add-ons can accurately predict the bookmarking actions of individual users—bookmarking clearly takes place in a Web browser and not, for example, in a word processor or a page-design program. But the situation is radically more complex and unpredictable for team integrations—which is why a point-and-click add-on would not likely serve the needs of the documentation team. A more thoughtful, customized solution for integration would be required, one that not only operationalizes the actions of viewing and visiting others' bookmarks but occurs in an existing environment that supports the activity of the team's work.

Delicious in Other Work Environments: Team Integration

Now that Delicious has been described as a service that can be conceptualized in terms of activity theory, we can consider how Delicious might be integrated in the less predictable work environments of multiple-user or team settings. Integrated with the browser, the options that individual users have for adding bookmarks to Delicious suggest a sort of continuum of textual coordination: On one end of the continuum is a highly disruptive use of the Delicious Web site that involves leaving one Web page for a new, labor-intensive page in order to perform error-prone actions such as cutting and pasting text. At the other end is the near-invisible integration of Delicious with the Web browser's native bookmarking capability, in which Delicious functions and displays content within the browser. This near-invisible, operationalized end of the continuum will be the focus of the team integration of Delicious.

My first attempts at designing team integration of Delicious stemmed from my desire to help graduate students share resources in a Web-design class. Students were frequently searching Google for XHTML references or CSS tutorials, which helped them solve particular design problems as they worked through the class projects. But once they had solved a problem or used a technique, they often simply closed the browser tab that was opened to the language reference or tutorial. Thus, the reference was unlikely to be remembered and never to be shared with other students, who might well have benefited from it. This invisible research seemed like important intellectual work that needed to be preserved and shared—just

as the team members working on documentation for the digital audio recorder might wish to preserve and share Web-available information of relevance to their project.

Although I could have set up a wiki for students to use for posting their Web-design finds, I wanted students to remain in control of their bookmarks. That is, I did not want their efforts to be bound up in a course Web site that I maintained, nor did I want them to lose their own bookmarks in a vast sea of everyone else's. Delicious would solve this problem—while also building students' literacies with SMAs. But the problems of using Delicious directly from its Web site in a class of a dozen or more students were obvious—and the same as the problems the documentation team would face: Expecting individuals to keep tabs on one another's accounts by visiting multiple pages on the Delicious Web site was out of the question because doing so would simply be too time-consuming. And if no one would be looking at anyone else's bookmarks, why save them on Delicious in the first place?

And because students often use their Delicious accounts in support of other activities and interests, I needed to ensure that students could leverage the flexibility of their accounts to support different activities so that they would never miss the chance to share resources of relevance to the rest of the class. I opted to ask students to use a memorable, simple tag for course-related resources: the course's three-letter, three-number designation, com530. In addition to the course designation, I encouraged students to use as many other tags as they deemed relevant for describing the resource. The documentation team too might use the model number of their digital audio recorder, perhaps dar2000, to help organize bookmarks related to that particular project.

Although students could have just monitored the page for the shared course-designation tag from all accounts across Delicious, I still wanted to avoid requiring students to visit even one more special page to browse one another's bookmarks. What I needed to do was to bring the different sets of bookmarks per account into a centralized place that students were already frequently visiting as part of their work flow for the class: the course Web site, which already had a fair amount of dynamic content such as agendas posted just before each class meeting and frequent updates and revisions to the calendar. Students, in other words, had operationalized the action of visiting the course Web site often, just as I imagine the documentation team visiting a centralized Web site or other shared tool. Integrating bookmarks as a sidebar on a frequently visited, centralized Web site would be an

obvious and effective choice if Delicious's services could be leveraged to achieve such a thing.

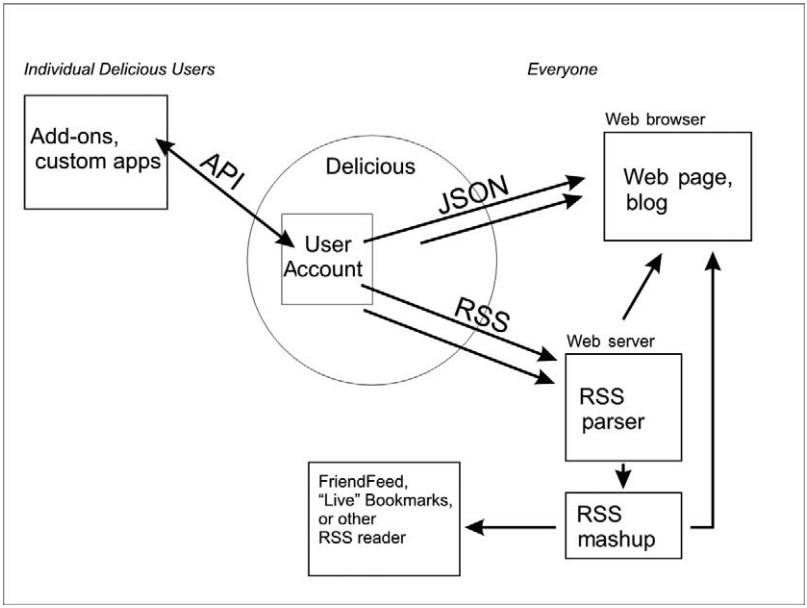
Delicious as an Open System: A Two-Way Street

Spinuzzi (2003) observed that "an open system is a centrally designed artifact, of course, but it exists as a nexus for worker's innovations," which he compared to an artificial coral reef's ability to provide "just enough stability to allow the ecology to flourish" (p. 205). To do that, a system must include "openings" that allow users to "link their own customized innovations with other innovations . . . both inside and outside the interface" (p. 212). Delicious is in many respects an open system—as evidenced by the proliferation of add-ons available to Firefox. I have mapped out some of Delicious's openings in Figure 2, which I will describe in some detail before applying them to activity theory.

None of the services illustrated in Figure 2 require a user to visit a URL at Delicious itself from within a Web browser although all are accessed behind the scenes via URLs (much like images are loaded into Web pages via URLs that readers may never see). Individual account holders can make use of the Delicious Application Programming Interface (API); indeed, without the API, the Del.icio.us Bookmarks add-on would not be possible. And although the JavaScript Object Notation (JSON) and RSS feeds provide a one-way street of data from Delicious to a remote application or reader, Delicious's API, like most APIs, is a two-way street, allowing users to retrieve and, in some cases, submit and modify information. Designer and digital media guru Maeda (2005) has suggested the following accessible metaphor for conceptualizing APIs:

> [An] API can be thought of as a company's internal telephone book with descriptions of key personnel's titles and job descriptions where you (the outsider) are free to call those people up to do your bidding. Some API's like Google's and Amazon's give you (the outsider) a limited number of phone calls to different sections of the company as a way to make sure that many outsiders can reliably get their phone calls answered . . . .
>
> You [access] a running computer *application* [via] an *interface* (the "phonebook") that one can *program* (the "phone call") externally.

Figure 2
Features of Delicious as an Open System



When a site chooses to make its API publicly available, "independent developers can spin decentralized services using centralized data" (Zeldman, 2007, p. 114). At the same time, as Figure 2 shows, the Delicious API is currently limited to accessing individual users' accounts. Delicious's help pages indicate that although development of the API is ongoing, at present the API is single-account focused and is therefore not suited to team-based integration. I mention the API at length here because many SMAs have APIs, and as they mature, APIs will likely play increasingly prominent roles in any SMA integration. But APIs are not limited to SMAs: The RSS parser (reader or interpreter) SimplePie, which I refer to in the examples in the next section, itself has an API that can figure prominently in the integration of Delicious into teamwork environments.

Whereas API is a broad concept, as in Maeda's (2005) metaphor, that may call on any number of computer languages, JSON and RSS are specific applications of particular languages: JSON is a "lightweight

data-interchange format ... based on a subset of the JavaScript Programming Language" (Json.org, n.d.), and RSS is a particular application of XML. Both JSON and RSS serve the same essential purpose of transmitting data in a structured form that machines can understand. JSON, as an object in JavaScript, can be used in JavaScript code directly on a Web page without being parsed on a server, as RSS is in Figure 2. My own preference is for XML-based RSS, but certainly arguments can be made in favor of JSON. For example, a technical communicator in a particular kind of work environment may find JSON to be a better solution—particularly if server-side processing is not available.

When technical communicators work to integrate an SMA into a work environment, whether through an API or with JSON or RSS, they necessarily come up against a wide variety of mediating artifacts—not to mention changes in workplace environments and technologies—that will shape their work and which must be investigated further in relation to actual practice. Kaptelinin and Nardi (2006) noted that mediating artifacts "usually reflect the experience of other people who tried to solve similar problems earlier and invented or modified the tool to make it more efficient and effective" (p. 70). Mediating tools have an important role in activity theory, which takes particular interest in the relationship of individuals to larger social groups. "Activity theory," Kaptelinin and Nardi (2006) explained, "sees all practice as the result of certain historical developments under certain conditions. Development continuously reforms and develops practice" (p. 71). The integrations that I have used in my course Web sites and that I imagine that the documentation team would use are not the kind that you can set up and forget about; they require a degree of maintenance not only to keep up with changes at Delicious (which, over the course of this writing, changed its RSS feed URLs) but to keep pace with the changing activity of technical communicators.

Still, as the discussion of APIs, JSON, and RSS implies, the practice of integrating SMAs is highly technical. Slattery (2007) has expressed concern that, in certain cases, technical writers "face problems being experts, not in what information should be organized and how, but only in the process of bringing it together." Furthermore, technical communicators whose expertise is "merely technological" may also find themselves in a situation in which "the ease of use of ... technology might threaten narrowly defined technical writing positions" (p. 323). But I would argue that changing needs and team activities, such as those supported by the use of SMAs, would demand more than a self-defeating, "merely technological" literacy, a position that Hart-Davidson's (2001) claim that writing itself is an information

technology would support. In terms of activity theory, the actions of writing, testing, and revising code may appear "merely technological"; indeed, we might ask if such actions should not be performed by IT staff instead. But technical communicators' ability to make explicit the connections between technological actions and larger individual or team activities should help them maintain, if not advance, their positions within organizations. Hart-Davidson noted that what information-management systems often fail at is handling "ad hoc workflows." The case of the team of technical communicators who worked on documentation for the digital audio recorder exemplifies how SMAs can be used to accommodate such work flows. The team members will move onto other projects and have new needs to support different activities, which will require them to revisit any sort of SMA integration in the future.

Nevertheless, technical communicators need to learn to write code as part of a larger project of building a new kind of digital literacy, one that moves beyond a user-only attitude toward technology. Technical communicators, as Hart-Davidson (2002) argued, need to think of themselves as designers. The use of add-ons in Firefox may be a point-and-click operation, but more advanced integration of SMAs for team use demands that technical communicators write code. Nardi (1993), writing about simple languages such as those for controlling spreadsheets, noted that "no matter how much designers and programmers try to anticipate and provide for what users will need, the effort always falls short because it is impossible to know in advance what may be needed," and therefore "end users should have the ability to create ... customizations, extensions, and applications" (p. 3). Fortunately, the key technologies for extending SMAs, including RSS feeds and APIs, are limited to "a carefully chosen set of task-specific operations that allow programming within a particular set of tasks" (p. 3). Because of this limitation, these technologies lack the "steep learning curve" of general programming languages (p. xii).

Integrating Delicious Into a Teamwork Environment: Customizing a Stream of Shared Bookmarks

Regardless of the ultimate destination—a Web page or a new, mashed-up RSS feed—of an integrated use of Delicious, all integrations share a number of key steps if they are based on Delicious or another SMA's use of RSS feeds. These key steps include locating all of the RSS feeds that you wish

Figure 3
Simplified Bookmark in Delicious's Rich Site Syndication (RSS) Feed

```
<item>
 <title>Customer Reviews: DAR 2000</title>
 <link>http://www.example-reviews.com/review-of-dar-2000.htm</link>
 <description>A set of reviews, mostly positive, of the
  Digital Audio Recorder 2000.</description>
 <category>dar2000</category>
 <category>customer-reviews</category>
 <category>positive-reviews</category>
</item>
```

to track, understanding the basic contents of these feeds, and identifying technology for parsing the feeds in order to extract relevant content and place it into a usable form (HTML, another RSS feed, even a database). Although I will keep the discussion as high-level as possible here, I have provided full, commented code examples that can be tested and viewed—and edited and used elsewhere—at http://karlstolley.com/code/ism/.

Identifying the Individual Streams of Bookmarks

When users work with RSS feeds from multiple Delicious accounts, the obvious first step that they need to take toward integrating these feeds is to learn the feeds' unique URLs through which individuals' bookmarks can be accessed. In our example, the documentation team could easily share their URL locations over e-mail. But anyone seeking to build an integration would be wise first to view the RSS feeds in a Web browser. This simple test ensures both that the feed exists (i.e., that someone has not mistyped a URL) and that the RSS feed uses valid XML. With Delicious, valid feeds are usually a given. But I have found that testing the feed first can save untold debugging hours further into the feed's integration. Unlike HTML documents, which Web browsers will valiantly attempt to display no matter how many errors they contain, XML documents that contain any errors will simply not display, and the browser will report an error.

RSS itself is a relatively straightforward application of XML. Figure 3 shows a somewhat simplified bookmark as it appears in Delicious's RSS feed. The "item" in Figure 3 contains the title of the bookmarked Web site, the link

to the site itself, a brief description, and several categories— Delicious tags—that classify it. In short, every bookmark posted to any team member's account also shows up in the team member's RSS feed. Keep in mind that the tags that make up this selection of the feed—title, link, description, category—are not unique to Delicious but are part of the RSS specification maintained at Harvard Law, so a technical communicator wishing to work with Flickr's or Twitter's RSS feeds would find the exact same set of core XML tags.

Parsing the Streams (RSS Feeds)

Understanding the basics of the XML tags found in RSS feeds, members of the documentation team would then need some sort of RSS parser to extract the information from the tags (see Figure 2); SimplePie is a popular open-source parser written in PHP (a recursive acronym for PHP Hypertext Preprocessor, which, as the name implies, is well suited to outputting HTML—the language of the hypertext Web). PHP is a server-side language: It runs on a Web server (unlike, e.g., JavaScript, which runs in a user's Web browser). SimplePie has a simple API for retrieving information from RSS feeds. So, for example, to retrieve the information from the RSS feed's title tag, a technical communicator would consult SimplePie's API reference, which includes the function for doing just that: "get_title()." This function streamlines the action of writing such code to almost the level of operations found in activity theory: If technical communicators are familiar with basic RSS tags, they already understand the SimplePie API. Similar, predictably named functions (or methods, as they are called within object-oriented programming) exist within SimplePie for retrieving information out of the other common tags found in RSS. The technical communicator would need to loop through this code for each team member's feed and then for each bookmark of each member's feed (see Hart-Davidson, 2005, for a related discussion of objects and views with respect to Web writing).

Merging the Streams to Build an HTML Page for Team Bookmarks

Once technical communicators on the documentation team have extracted the information of interest to them, they can easily place the information into any manner of HTML tags, such as those for unordered lists, and store these tags within the code of the team's project Web site. What appears on the integrated site, then, is simply a list of all of the team

members, with each name followed by a number of their most recent posts to their Delicious accounts. The team members can view and interact with the items, which display as any other simple HTML would: The hyperlinked title of the bookmark, its description, and tags appear on the page, allowing members of the team to follow through on their operationalized behavior of clicking on links that interest them. But team members will not be any the wiser, unless they are in on the code behind the integration, of the journey the bookmark has made from Delicious, through the RSS feed, into SimplePie, and finally into HTML and their browser window.

Redirecting the Stream by Using RSS Feed Readers to View the Team Bookmarks

But suppose that the documentation team does not have a centralized project Web site or is unable to do the sorts of customization to the site that is required in the previous example. Provided that the team members at least have server access to post a small PHP script, rather than adding HTML to a Web page, they could instead deliver a mashup of their bookmarks in RSS. This mashup would be a brand new feed that could be viewed in any RSS reader that a team member might prefer: The "Live Bookmarks" reader in Firefox, or perhaps Google's FriendFeed or Google Reader. The team member performing the integration, then, would instruct PHP to prepare the file as XML and rewrite the code to use tags from RSS instead of HTML.

Some might find this last example confusing and redundant: Why take an RSS feed and make it into yet another RSS feed? The reason, of course, is that the new feed includes items from the entire team's collection of bookmarks, allowing individual members of the documentation team to browse the collection in any RSS reader of their choosing. If someone on the team already uses a feed reader to track blogs or even stock prices, this integration might already be an operationalized action. And given that so many RSS readers are browser based, that team member's experience of browsing to potential sites of interest identified by colleagues will be seamless—and again, will occur without ever visiting Delicious at its own Web site.

Enhancing the Stream by Adding Functionality

Going a step further, a team member could even develop a script that is based on the same PHP and SimplePie code but that captures all of the team's bookmarks for a project and stores them in a database that the team

controls. Once the bookmarks are stored in a local database, a blog-style commenting or rating function could easily be added, allowing team members an even more socially intensive, integrated experience of sorting through and working with one another's bookmarks.

Conclusion

As Zeldman (2007) has remarked about Web services in general, the integrations I have presented here "are like the one-reel silent movies of the late nineteenth century: interesting in themselves, explosive in what they portend" (p. 114). But it is key to not focus too much on specific mediating artifacts—SMAs or otherwise—and to always be mindful of the broader activity that mediating artifacts are intended to support. Slattery's (2005) observations about "textual coordination" are instructive in this sense: The actions that make up textual coordination are not ends in themselves but are ad hoc strategies and work-management techniques that are always intended to advance some broader activity. The Delicious browser add-ons, for example, are somewhat astounding in terms of how they enable individual users to store and share their bookmarks through the Delicious service. But these add-ons are only useful if they unobtrusively support the individual user's work activity.

The same importance of usefulness applies to the different ways I have suggested that technical communicators may integrate Delicious into other work environments for team use. In Spinuzzi's (2003) imagined open system for highway workers to share traffic data, one of the most important features is that workers not only can create their own mashups to share data and information but can share the little snippets of code that enable them to complete their tasks and advance their own goals. Any successful, meaningful, and participatory integration of Delicious or any other SMA must provide a means for users to post their own code snippets and add-ons for use by others within their organizations—if not the broader world—just as developers of Delicious add-ons have done by posting their work at the Mozilla Web site.

Ultimately, I can imagine a future in which technical communicators are able not only to combine and repurpose RSS feeds but also to learn to command the APIs of any SMA that would support their work activity. Such technical communicators could advance even further to integrate SMAs with open-source browsers such as Firefox and build their own personal or team- or organization-specific add-ons for a given SMA. Such add-

ons might fit even more closely with the actions of communicators within their organization—and the activities those actions support. As I have shown here, the technical and theoretical foundations for realizing such activity are already in place. Technical communicators need only to make purposeful, activity-driven use of them.

References

Amidon, S., & Blythe, S. (2008). Wrestling with Proteus: Tales of communication managers in a changing economy. *Journal of Business and Technical Communication*, *22*, 5-37.

Braly, M. D., & Froh, G. B. (2006, November). *Social bookmarking in the enterprise*. Poster session presented at the 17th Annual ASIS&T SIG/CR Classification Research Workshop, Austin, TX. Retrieved August 31, 2008, from http://dlist.sir.arizona.edu/1665/

Czerwinski, M., Horvitz, E., & Wilhite, S. (2004). A diary study of task switching and interruptions. In *CHI'04: Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (pp. 175-182). New York: ACM Press.

Hart-Davidson, B. (2001). On writing, technical communication, and information technology: The core competencies of technical communication. *Technical Communication*, *48*, 145-155.

Hart-Davidson, B. (2002). Turning reflections into technology: Leveraging theory and research in the design of communication software. In *Proceedings of the International Professional Communication Conference* (pp. 455-467). Portland, OR: IEEE.

Hart-Davidson, B. (2005). Shaping texts that transform: Toward a rhetoric of objects, relationships, and views. In C. Lipson & M. Day (Eds.), *Technical communication and the World Wide Web* (pp. 27-42). Mahwah, NJ: Erlbaum.

Hutchings, D. R., Smith, G., Meyers, B., Czerwinski, M., & Robertson, G. (2004). Display space usage and window management operation comparisons between single monitor and multiple monitor users. In *AVI'04: Proceedings of the Working Conference on Advanced Visual Interfaces* (pp. 32-39). New York: ACM Press.

Json.org. (n.d.). *Introducing JSON*. Retrieved September 1, 2008, from http://json.org

Kaptelinin, V. (1996). Computer-mediated activity: Functional organs in social and developmental contexts. In B. A. Nardi (Ed.), *Context and consciousness: Activity theory and human-computer interaction* (pp. 45-68). Cambridge, MA: MIT Press.

Kaptelinin, V., & Nardi, B. A. (2006). *Acting with technology: Activity theory and interaction design.* Cambridge, MA: MIT Press.

Maeda, J. (2005, May 26). API decoder ring. *Maeda's SIMPLICITY*. Retrieved September 3, 2008, from http://weblogs.media.mit.edu/SIMPLICITY/archives/000220.html

Millen, D., Feinberg, J., & Kerr, B. (2005). Social bookmarking in the enterprise [Electronic version]. *ACM Queue: Architecting Tomorrow's Computing, 3*. Retrieved August 31, 2008, from http://www.acmqueue.com/modules.php?name=Content&pa=showpage&pid=344.

Nardi, B. A. (1993). *A small matter of programming: Perspectives on end user computing.* Cambridge, MA: MIT Press.

Slattery, S. (2005). Technical writing as textual coordination: An argument for the value of writers' skill with information technology. *Technical Communication*, *52*, 353-360.

Slattery, S. (2007). Undistributing work through writing: How technical writers manage texts in complex information environments. *Technical Communication Quarterly*, *16*, 311-325.

Spinuzzi, C. (2003). *Tracing genres through organizations: A sociocultural approach to information design.* Cambridge, MA: MIT Press.

Spinuzzi, C., Hart-Davidson, B., & Zachry, M. (2006). Chains and ecologies: Methodological notes toward a communicative-mediational model of technologically mediated writing. In *Proceedings of the 24th Annual ACM International Conference on Design of Communication* (pp. 43-50). Myrtle Beach, SC: ACM.

Weinberger, D. (2007). *Everything is miscellaneous: The power of the new digital disorder.* New York: Holt–Times.

Zeldman, J. (2007). *Designing with Web standards* (2nd ed.). Berkeley, CA: Pearson Education–New Riders.

Karl Stolley is an assistant professor of technical communication at Illinois Institute of Technology in Chicago. He researches and teaches the use of open formats and standards and open-source software to design Web applications, content-management systems, and humanized interfaces. His blog, *source/literacy*, and professional Web site are located at http://karlstolley.com.