"Sanity Check"
John Unsworth

Summary remarks at the end of day 1 of the Peer Review Meeting at the American Geophysical Union, Washington DC, May 15-16, 2014. Sponsored by the Sloan Foundation.

As Kathleen began the day by suggesting, we used to have a system where in order to publish (in an academic discipline) one had to go through peer review, because opportunities to publish were scarce. We are now in a world where anyone can publish, and where the scarce resource is not opportunity but attention, so the most important form of peer-review may be retweeting, blogging, and clicks on links, academic or not. But still, we feel we need peer review. Or at least we feel that the other guy needs it. And he should publish open access. And expose his data. And keep it available forever.

From today's discussion I have also learned that peer review isn't one thing: it is many--but all of them can be thought of as involving some form of annotation. And I have learned that there are projects represented here today that aim to provide annotation in the wild, as it were—for anything on the web—and others that provide it in more gated communities, where the environment is relatively controlled. I'm sure there is a place for each type of project in the future, but I think the condition to which something like the Open Annotation Collaboration aspires is annotation in the wild, across document types, repositories, browsers, operating systems, etc.

The idea of holding a conference on annotation and peer review suggests that they have something to do with one another, and the obvious conclusion is that annotation is a method for doing peer review, or peer review is a form of annotation. But they are separate: annotation is a way of commenting on all or part of an object of attention, and peer review is a particular type of comment, with a particular gate-keeping or attention-directing purpose. For annotation as a mode of peer review, one can imagine the gated community serving the main purpose—after all, we are already accustomed to things like the Open Journal System, or ScholarOne, that require us to log in to a particular system in order to do our reviewing. Yet we persist in talking about annotations crossing platforms, shared between publishers, following documents through different stages of production and revision.

When we think of annotation as something for which we could build tools, tools that could be used in peer review, we confront the fact that a tool implies a model of a task, and therefore the tool implies models of the objects on which the tool will be used, and models of the process of using the tool. And that's where things start to get difficult.

Is annotation of scientific text the same as annotation of poetic text? Is annotation of text the same as annotation of music, or of film? Of course not: but we can perhaps imagine a conceptual model of annotation that is capacious enough to be applied to different data types, and to respond to different purposes. So, let's assume, per Doug Schepers, that we have a conceptual model, an abstract data model, a shared vocabulary, some agreed-upon serializations, an HTTP API, a client-side API, robust link anchoring and a suite of use-cases for an annotation tool. Now all we need is to get people within particular domains to use that tool in some consistent way, and, as Peter Brantley pointed out, we'll see all kinds of benefits. We'll be able to provide credit for the work that goes into peer review; we'll be able to increase the value of peer review for authors; we'll be able to link to precise parts of articles, and navigate those links bi-directionally.

Ah, the bi-directional link. The very idea is itself a bi-directional link, and it links the idea of the future we're trying to imagine, with the past we've nearly forgotten. Dan Whaley mentioned the fact that Marc Andreesen had built annotation into the early versions of Mosaic, but that he abandoned that feature set because the links proved too brittle, the storage too centralized and not scalable, the whole business too open to abuse, and lacking an agreed upon ontology or data model. So, let's look backwards for a bit, back 18 years, to 1996, when Clay Shirky wrote a piece for ACM's net_worker, called "In Praise of Evolvable Systems, or Why something as poorly designed as the Web became The Next Big Thing, and what that means for the future"
(http://www.shirky.com/writings/herecomeseverybody/evolve.html)

"If it were April Fool's Day," Shirky wrote, "and you wanted to design a 'Novelty Protocol' to slip by the Internet Engineering Task Force as a joke,

it might look something like the Web:  The server would use neither a persistent connection nor a store-and-forward model, thus giving it all the worst features of both telnet and e-mail.  The server's primary method of extensibility would require spawning external processes, thus ensuring both security risks and unpredictable load.
[...]
The hypertext model would ignore all serious theoretical work on hypertext to date. In particular, all hypertext links would be one-directional, thus making it impossible to move or delete a piece of data without ensuring that some unknown number of pointers around the world would silently fail. The tag set would be absurdly polluted and user-extensible with no central coordination and no consistency in implementation. As a bonus, many elements would perform conflicting functions as logical and visual layout elements.

HTTP and HTML are the Whoopee Cushion and Joy Buzzer of Internet protocols, only comprehensible as elaborate practical jokes. For anyone who has tried to accomplish anything serious on the Web, it's pretty obvious that of the various implementations of a worldwide hypertext protocol, we have the worst one possible.

Except, of course, for all the others.

[by which I imagine Shirky means Xanadu, Intermedia, HES, NLS, KMS, FRESS, DPS, Hyperties, Symbolics Document Examiner, even the commercially successful Hypercard, and how many more? See Daniel Yule and Jamie Blustein, "Of Hoverboards and Hypertext," DOI: 10.1007/978-3-642-39229-0, http://web.cs.dal.ca/~yule/pdf/hypertext.pdf]

Three Rules For Evolvable Systems

Evolvable systems -- those that proceed not under the sole direction of one centralized design authority but by being adapted and extended in a thousand small ways in a thousand places at once -- have three main characteristics that are germane to their eventual victories over strong, centrally designed protocols.

1. Only solutions that produce partial results when partially implemented can succeed. The network is littered with ideas that would have worked had everybody adopted them. Evolvable systems begin partially working right away and then grow, rather than needing to be perfected and frozen. Think VMS vs. Unix, cc:Mail vs. RFC-822, Token Ring vs. Ethernet.

2. What is, is wrong. Because evolvable systems have always been adapted to earlier conditions and are always being further adapted to present conditions, they are always behind the times. No evolving protocol is ever perfectly in sync with the challenges it faces.

3. Finally, Orgel's Rule, named for the evolutionary biologist Leslie Orgel -- "Evolution is cleverer than you are".  As with the list of the Web's obvious deficiencies above, it is easy to point out what is wrong with any evolvable system at any point in its life. No one seeing Lotus Notes and the NCSA server side-by-side in 1994 could doubt that Lotus had the superior technology; ditto ActiveX vs. Java or Marimba vs. HTTP. However, the ability to understand what is missing at any given moment does not mean that one person or a small central group can design a better system in the long haul.

Centrally designed protocols start out strong and improve logarithmically. Evolvable protocols start out weak and improve exponentially. It's dinosaurs vs. mammals, and the mammals win every time. The Web is not the perfect hypertext protocol, just the best one that's also currently practical. Infrastructure built on evolvable protocols will always be partially incomplete, partially wrong and ultimately better designed than its competition."

So, on May 15, 2014, I would encourage this group (or those among you who aspire to annotation in the wild) to think of designing networked annotation as an evolvable system, one that could begin working immediately, albeit imperfectly; one that works even if everyone doesn't agree on how to use it; one that works even if it doesn't solve some of the problems that are clearly in its queue; one that can be extended in a decentralized way, that is fault tolerant, that is incomplete to begin with, and never really up to date, but that nonetheless delivers partial results when partially implemented, and does something obviously useful right away.

It seems to me that the core definition of 'annotation' is a comment connected to an object by a pointer.  In a perfect world, we'd need the document owner to allow us to place a "robust anchor" in the object-document in order for this to work well—but let's assume that's not possible. What would you do then?  You'd either point in such a way that the pointer itself says what it is pointing at, so you'd know if the object had changed, or the system would capture a copy of the object-document (or the relevant part of it) and store it where it could be kept in its original state, a snapshot, as it were, perhaps including a document-level reference to the original source.  Terrible, I know, from all sorts of angles, but partly workable right off the bat.

With respect to peer review, as with the topic of data sharing, our imaginary evolvable system still doesn't address the all-important question of motivation (for the author, for the editor, for the annotator).  On the other hand, it suggests a peer-review scenario where the author wouldn't have to adopt new authoring tools, the journal wouldn't have to adopt new workflows, and the annotator could annotate and move on.  And for those who were motivated, the nature of each pointer and its object would tell us something about the unit of annotation, in different domains or different use cases.  Maybe the tool could be smart enough to recognize a Document Object Model if one were present, and could use the elements of the DOM as hooks on which to hang JSON-LD or other forms of context.  Of course, in a given document, the DOM might change depending on how you load the document, or how you arrive at it.  In that case, having the snapshot to fall back on wouldn't be a bad thing.  Sort of like Dan's Hypothes.is demo, which cleverly keeps the original text that was the object of an annotation, even after the text has been changed, as a result of the author responding to an annotation.

Around the edges of all this talk about systems and software lurks the fact that we are also talking about a complex human behavior.  Although we talk about peer review as though it were a single thing, it becomes clear when you look at particular examples that no two peer review processes are the same, and that there are major differences across domains (humanities, social science, natural science, mathematics, data science, etc.).  I expect, given tomorrow's program, that we will be hearing more about that the scholar's view of all this then.  Across disciplines and domains, the objects reviewed are different; the goals of the review are different; the process is different; the outcomes are different; the language is different, and so on.  Layer on to that cultural differences, commercial differences, historical differences, personal differences, and we find ourselves in the position of building a tool to accomplish a task without a widely accepted definition, on research outputs of very different types, for widely divergent purposes, in a rapidly changing technical environment.  All of which leads me back to the attraction of the evolvable system--not finished, but extensible, not a perfect fit for any particular peer review process, but useful in many different scenarios, and—like these remarks—leaving no one fully satisfied.

Thank you.