

Making a Contribution: Modularity, Integration and Collaboration Between Tools in *Pliny*

John Bradley (john.bradley@kcl.ac.uk)

Centre for Computing in the Humanities
King's College London

Computing tools have been an issue since the foundation of Humanities Computing, and building modular tools that work together has been recognised as important since at least the CETH meetings in 1995. Geoffrey Rockwell and I first raised issues of modularity in our paper given at the Canadian Learned Societies Conference in June 1992 entitled “Towards new Research Tools in Computer-Assisted Text Analysis”. Our proposed tool framework combined data-flow tool modularity with the ability to create pages that mixed scholarly writing with interactive elements like in a *Mathematica* “Notebook”. Of course the WWW was sweeping all this away by 1995 and when we took a paper (Bradley, Rockwell 1995) to the CETH “tools” meeting we were merging our modular view with the WWW as it was then emerging.

In our community “modularity” often comes down to the sharing of file formats. Separate tools that can all read the same file format can all operate on common data, and therefore can contribute their particular facilities to the task at hand. File- or Unix-style modularity is a part of what I called the “transformation model” of computing (Bradley 2005), and provides a powerful approach to manipulating data. TuStep (described in Ott 2000) is a splendid example. The model holds a strong attraction within the Digital Humanities community – even extraordinarily creative projects as TAPoR’s text analysis portal (TAPoR 2006) or the Nora project (Nora 2006) are, in fact, primarily based on this. However, thinking of modularity primarily in those terms limits our views to those of computing about 25 years ago – before the advent of the graphical user interface (GUI).

The GUI radically changed the way we think about computing, and even today, more than 25 years later, its significance continues to reverberate. It is surely true to say that most humanists have the GUI as a model driving how they think of their computer. The impact has been twofold:

- First, users now expect that the computer screen will allow them to directly manipulate materials that interest them. Think of the words in a word processor.
- Second, interaction between tools needs to be possible not only between files, but also on the screen as well. Users expect to be able to incorporate parts of spreadsheets inside word documents, for example.

To reflect this broader sense of collaboration between tools, we need some different language. In this paper I will use the word *integration* for that aspect of tool collaboration that focuses on GUI issues. The use of *integration* might make us also think of the integrative nature of humanities scholarship – an intentional parallel.

Integration in the GUI presents challenges. First, the development of tools allowing for direct manipulation of objects on the screen is more complex and costly than resolving file-sharing issues. Furthermore, if independent tools are to interact on the screen – elements maintained by potentially different tools sharing screen space – then they must operate in a computing framework that makes this kind of thing practical.

Pliny (Pliny 2006) has been developed precisely to draw attention to these two issues and to encourage some thinking about cooperating tools beyond file-oriented “modularity”. It both significantly broadens how computers can support humanities research and suggests a much richer and more acceptable interface to those tools that might well attract a larger number of scholars. *Pliny* tackles this in two ways. First, it supports annotation and note taking – functions central to several aspects of humanities scholarship and which I believe have been largely neglected by the DH community – and, second, it is built using a framework called Eclipse (Eclipse 2006) which is deliberately created to support GUI-level interaction between tools of the kind I mentioned above. Eclipse also supports rich collaboration between tools developed by independent developers.

Personal annotation and note-taking compels us to think about tool integration because it is widespread in humanities scholarship and runs across all kinds of scholarly work. Scholars write notes to record their reactions to not only books they read, but also web pages they view. Furthermore, if they have tools that do (say) text analysis, they would probably want to record notes about that as well. The provision of note-taking within a *particular* website is insufficient for personal note-taking (although it might well fulfil a useful need supporting public comment about the material the website offers) because most scholars work across a broad range of materials, and their note taking capability must reflect this. Notes from a book will at some point need to be brought in contact with notes about, say, an online archive or a journal, or about findings from the text analysis tool. Personal scholarly note taking integrates by its very nature.

Personal annotation also draws our attention back to the *software application* as the context in which tools can and should be built, rather than thinking of the browser as the context for tool delivery. It is surprising how difficult it is to make this point to those in the Digital Humanities. Of course part of the reason is that XML, one of the key technologies that fuels some portion of the DH community, has been developed specifically to work in the context of the WWW. However, if we wish to develop a more broadly based humanities community who use technology in more sophisticated ways to support their research we need to broaden our focus beyond the WWW. Providing scholarly resources on the WWW supports scholars, of course, and there remains some excitement that they can get materials readily right to their desktop. However, as I argued in Bradley 2005, once on that desktop all they can do with them is read them on the screen or print them out – a webpage, even one designed with all the clever parts of CSS and AJAX, allows only manipulation within its own page context. It contributes little to the integrative aspect of personal scholarship which must, by its very nature, often bring materials together from disparate sources.

Software Applications work with personal materials. Surely this is a central element of humanities scholarship. A word processor creates materials that *belong* to the researcher. Until web browsers can be used to create materials that are stored as personal data – on the user's own machine – they cannot replace applications (see a similar assertion from a technical perspective in Charland 2005). Even the XML folk – part of the community driving a WWW view of humanities computing because of XML's compatibility with website creation – use an application such as Oxygen to create XML materials in the first place. For the same reason that the creation of an XML file involves an editor rather than a web browser, the creation of personal notes – especially those gathered from reactions across a great range of sources – requires a personal application. Furthermore, more than one study into computing and humanities scholarship (see both Brockman *et al* and Siemens *et al* 2004) has found that scholars have tried to apply inappropriate applications such as word processors to support this need, and have not been very satisfied with the result.

Applications can be built in several different frameworks. *Pliny* is written in Java using the Eclipse framework rather than the one provided by Sun to create desktop tools. I didn't choose Eclipse because it was easy for me to use it – indeed I had to learn it from the beginning during the past year or so. I choose Eclipse because it is designed specifically to support the integration of tools developed by separate developers. Eclipse provides a way to share out screen space between windows managed by different tools. Furthermore, it also provides mechanisms for elements from separate tools to share the same window (called “making a contribution” in Eclipse). Tools built with the Eclipse plug-in model need not operate in isolation

from other related tools. A text analysis tool built in Eclipse could use *Pliny* elements to allow users to record notes while using the TA tool, and the notes that were recorded in this way would also be visible in the context of notes created with other materials – say from reading a book, or annotating a web page.

So, in this presentation I have an impossible task. First, I am promoting the idea that we focus more on application development than web development, a technically more demanding activity, and one that goes against the grain of much work in DH over the last decade. Second, for those who might subscribe to this idea, I am promoting that we build our tools according to the Eclipse model rather than the more widely understood Sun/Java framework. Perhaps I won't convince anyone here. However, I believe that unless we start to think of tools in the context of applications, as our scholarly community does, and unless we start to think more seriously of tool building in the context of GUI integration in addition to data sharing, we will never get the attention of most scholars in the humanities.

Bibliography

Bradley, John. "What You (Fore)see is What You Get: Thinking about Usage Paradigms for Computer Assisted Text Analysis." *Text Technology* 14.2 (2005): 1-19. Accessed 2006-09-01. <http://texttechnology.mcmaster.ca/pdf/vol14_2/bradley14-2.pdf>

Bradley, John, and Geoffrey Rockwell. "Towards New Research Tools in Computer-Assisted Text Analysis." Paper presented at The Canadian Learned Societies Conference, June 1992. 1992. <<http://www.cch.kcl.ac.uk/legacy/staff/jdb/papers/learneds.html>>

Bradley, John, and Geoffrey Rockwell. "The Components of a System for Computer Assisted Text Analysis." Paper presented at the CETH Workshop on Future Text Analysis Tools, October 1995. 1995. <<http://www.cch.kcl.ac.uk/legacy/staff/jdb/papers/ceth95.html>>

Brockman, William S., Laura Neumann, Carole L. Palmer, and Tonya J. Tidline. *Scholarly Work in the Humanities and the Evolving Information Environment*. Washington, D.C.: Digital Library Federation and Council on Library and Information Resources, 2001.

Charland, Andre. "Will Ajax Replace the Desktop?" *developer.com*. 2005. Accessed 2006-10-01. <http://www.developer.com/design/article.php/10925_3574116_1>

Eclipse. 2006. Accessed 2006-10-01. <<http://www.eclipse.org/>>

Nora. 2006. Accessed 2006-10-01. <<http://www.noraproject.org/>>

Ott, Wilhelm. "Strategies and Tools for Textual Scholarship: the Tübingen System of Text Processing Programs (TUSTEP)." *Literary & Linguistic Computing* 15.1 (2000): 93-108.

Pliny. 2006. Accessed 2006-10-01. <<http://pliny.cch.kcl.ac.uk/>>

Siemens, Ray, Elaine Toms, Stéfan Sinclair, Geoffrey Rockwell, and . "The Humanities Scholar in the Twenty-first Century: How Research is Done and What Support is Needed." *ALLC/ACH 2004 Conference Abstracts*. Göteborg: Göteborg University, 2004.

TAPoR: *Text Analysis Portal for Research*. 2006. <<http://test-tapor.mcmaster.ca/portal/portal>>