Dozens of Little Radio Stations: Getting Technologies Talking in the MONK Workbench

Andrew McDonald, Amit Kumar, Matt Bouchard, Alejandro Giacometti, Matt Patey, Milena Radzikowska, Piotr Michura, Carlos Fiorentino, Stan Ruecker, Catherine Plaisant, and Stefan Sinclair

The Metadata Open New Knowledge (MONK) project is an attempt by seven universities and about 35 researchers to make sophisticated technologies for analyzing and visualizing texts available to literary scholars (Sinclair et al. 2008). Funded by the Mellon Foundation, the project is being led by John Unsworth of the University of Illinois at Urbana-Champaign and Martin Mueller from Northwestern University. There are a wide range of tools and techniques available, and the goal of MONK is to bring at least some of them into a friendly environment that will encourage what Steve Ramsay (2003) has called "algorithmic criticism," the goal of which is to facilitate the iterative process of exploration, synthesis, pattern finding, and hypothesis formulation. This differs from hypothesis testing, where the system would produce some definitive result, and is instead much more like the familiar hermeneutic process of spotting something potentially interesting, and using a number of tools to follow up on it.

To make this possible, we have been designing the MONK workbench (Figure 1), which is an online system that combines access to digital documents with tools for manipulating them, including visualizations for examining the results at each stage. The workbench also sequences the tools into toolsets that can help users work through a process of investigation. From a technology perspective, we wanted the MONK workbench to be lightweight, capable of running in any web browser, but containing tools that have been written in a variety of platforms. As a result, we have programmed the workbench in EXTJS, but it contains tools written in Javascript, Flash, and Java, and it also incorporates tools derived from third-party APIs, such as Google Charts. The workbench communicates to Java on the server via a proxy layer that returns XML. The proxy was an important component, since it isolated, at least to some extent, the interface development cell from the cells producing the analytics and data.

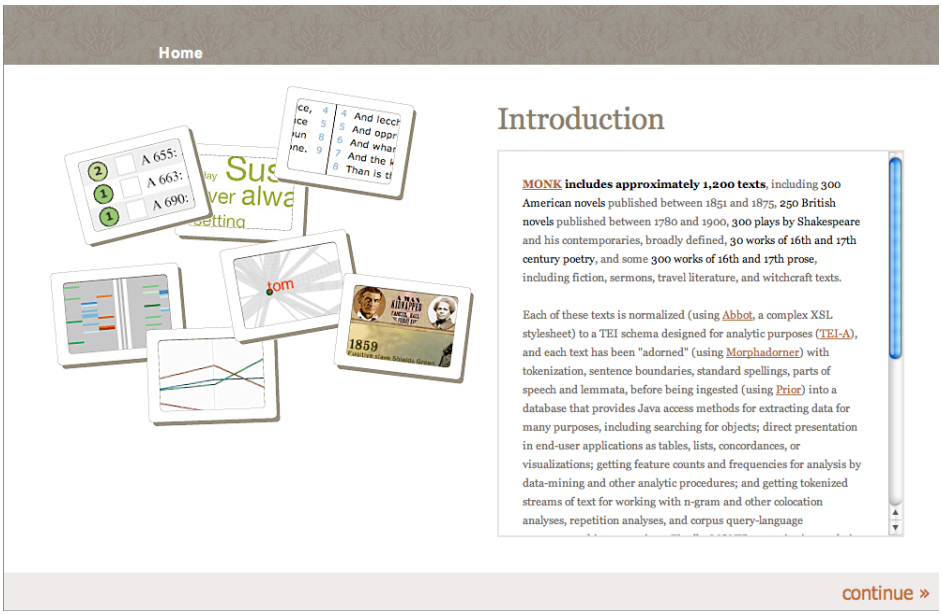

Figure 1. The MONK workbench is designed as a kind of pegboard that holds tools that are combined into toolsets. It is intended for literary scholars interested in using sophisticated technologies to study and visualize texts.

In this paper, we discuss the design of the MONK workbench, and focus on the nearly insurmountable problem of creating a Javascript-based pegboard that can include tools written using a variety of

Javascript libraries, as well as Flash and Java. In theory, each of these technologies can produce tools that can talk to each other, but in the context of the MONK workbench, that conversation needs to be fairly loquacious, because what the user does in one tool may influence what is going on in other tools. The strategy was to produce in the workbench itself a process analogous to the one used in object-oriented programming, where objects transmit and receive broadcasts and respond to them appropriately. In the case of the workbench, however, this broadcast system would work not within the context of a single programming language, but instead between tools.

A basic example is the search function, which resides in the structure of the workbench, but is heard and responded to by whichever tool has the focus in the currently open toolset. Tools need to know how to react to a search broadcast, so that a tool for collection browsing will respond with a list of works, while the tool that displays a work for reading will respond by highlighting the search term.

More complicated are the interactions between tools developed in different platforms. The Javascript listener and broadcaster built into the workbench is not very compatible with tools written in Flash, and is even less compatible with the ones that were coded using Java (the existing browser-based communication protocols between Javascript and Java are currently quite flakey). The result is that we are having to experiment with a parallel Java-based echo station that resides on the server and handles the broadcast and reception for Java tools, while also updating the client side station written in Javascript in the workbench. Since all of the Javascript tools are already listening and broadcasting with the workbench, the principle is the same, except now we have a variety of these little radio stations all talking with each other, between components as mediated by the workbench, between the workbench and its tools, and between the server station, the workbench, and the tools. In some, there is a lot of discussion going on behind the scenes in the MONK workbench. We are also experimenting with similar client-server communication strategies for providing progress reports to users about longer or more complex processes (such as those being run by remote high performance clusters).

What this means for the user is that the MONK tools should not be understood as silos, but instead what is going in inside one MONK tool should effect what is going on in related tools from the same toolset. For instance, someone using the faceted collection browser (written in Javascript) can also select documents one at a time for reading (Javascript again), and may in addition want to see a visualization of how the characteristic vocabulary of one of the documents compares to a baseline from a set of other documents (written in Flash). Someone using the Mandala browser (written in Java) to create a complex visual subset of documents should also be able to seamlessly call up the same reader tool (in Javascript) to read individual works or parts of works, then look at a clustering diagram (back in Java).

That any of this works is a small miracle, and is a tribute to the genius and hard work of the many scholars and programmers who are contributing to the project. However, it is also important to recognize that despite everything we are able to manage behind the scenes, there are at least three kinds of tools in the MONK workbench, based on how the tools were written. The native Javascript tools, including the workbench itself, should run in every browser. The Flash-based tools should work in any browser that has the Flash plugin, which is all browsers from the past few years. The Java-based tools will run in browsers that have the right flavour of Java, which is a smaller subset of available browsers. We are not really at a point in the culture of computing where a resource like the MONK workbench can be created, but we are proud to say that we have been trying hard to create it anyway, and hope that some of the strategies we have been working with will prove useful, as will the workbench itself, to the larger academic community.

References

Ramsay, Stephen. "Toward an Algorithmic Criticism." *Literary and Linguistic Computing*. *18*(2), 2003.

Sinclair, Stéfan, Andrew Macdonald, Matthew Bouchard, Mike Plouffe, Alejandro Giacometti, Amit Kumar, Milena Radzikowska, Stan Ruecker, Piotr Michura, Carlos Fiorentino, Matthew Kirschenbaum, and Catherine Plaisant. "Late Nights at the Scriptorium: Interim Results from the Interface Cell of the MONK Project." Paper presented at the Society for Digital Humanities/ Société pour l'étude des médias interactifs annual conference at the 2008 Congress of the Social Sciences and Humanities, University of British Columbia. June 2-3, 2008.