White Paper Report

Report ID: 98552

Application Number: HD5022407

Project Director: Douglas Reside (dreside@umd.edu)

Institution: University of Maryland, College Park

Reporting Period: 9/1/2007-8/31/2008

Report Due: 11/30/2008

Date Submitted: 9/15/2010

NEH Progress Report for AXE

Project # HD5022407

Term length: 09/01/2007 - 08/31/2008

The Maryland Institute for Technology in the Humanities (MITH) published the beta release of the Ajax XML Encoder (AXE), sponsored by a Digital Humanities Startup grant from the National Endowment for the Humanities on the MITH website on September 1, 2008. AXE is a web-based annotation tool capable of identifying "regions of interest" in video, audio, and image files located anywhere on the internet, and encoding these regions in an XML file. It can also be used to generate XML documents from texts located on the web. A functional prototype of the code is available at <http://mith.umd.edu/AXE>, and the source code is available in a subversion repository at <svn://khelone.umd.edu/AXE>.

Outcomes

In the short time since the project's alpha release in September of 2008, AXE has generated almost overwhelming interest in the digital humanities community. It is currently an important part of at least five new grants currently under consideration at the National Endowment for the Humanities (including several from institutions such as CUNY and the University of Kentucky originally unconnected with MITH), and portions of the code have been ported to the NEH-JISC funded Shakespeare Quartos Archive in order to permit annotation of the images and text in the collection. Additionally, a demonstration of the tool in a poster session at the Text Encoding Initiative's member's meeting at King's College, London in November, 2008 generated a great deal of discussion and it seems likely more projects will use the tool in the near future.

Perhaps most importantly, AXE recently caught the eye of the Mellon foundation and it seems likely they will soon be funding its future development. In the early fall of 2008, the AXE team spoke with programmers at the Center for History and New Media at George Mason University about the possibility of incorporating the AXE software library into their Zotero citation tool. Soon after, Chris Mackie and Ira Fuchs introduced the teams to Herbert Van de Sompel and a team currently current project. It was clear that the AXE software library, with its ability to deeply link video, audio, text, and images located at remote URIs would be a useful test tool for the annotation schema the team planned to develop, and Zotero's immense user base would prove invaluable for both evaluation and promotion of the schema.

Lessons Learned

Although, in many ways, the development year of AXE has been successful beyond the hopes of even the development team, the startup period has not been free of mistakes and miscalculations. In the interest of the Digital Humanities community, we want to document the process to help others avoid our missteps and follow where we were successful. Like many projects, the biggest mistake we made was underestimating the time it would take to go from a demonstration version of the software, useful for a programmer able to catch and fix small bugs as they arise, to a robust, debugged version of the code usable by even a technophobic humanist. This step, which seemed at first a small one, ultimately consumed the vast majority of the resources committed to the grant and required additional investment from MITH's normal operating budget. The lesson here is a familiar one but one which probably deserves yet another repetition—overestimate time requirements, anything that seems like it will take a few days will probably take a few weeks, and anything that seems like it should a few weeks will takes months.

Additionally, we would caution any future applicants to carefully consider the use of graduate student programmers. Although graduate students are often highly motivated and talented (as the AXE graduate student certainly was), they tend to be over-committed, inexperienced, and often not the most economical investment despite what may initially seem to be a cheap hourly rate. Additionally, our University, like many others, recently began requiring departments using graduate students funded by grant dollars to pay not just the student's salary, but also their benefits and tuition remission (around a total of $40,000 in our case). This meant the approximately $20,000 available to AXE after the overhead costs were taken could pay for only about 10 hours of a graduate student's time. An entry-level half-time programmer could be had for almost the same amount. Further, 10 hours of research assistant work is likely not the most productive or focused of a graduate student's already busy intellectual life. If the student's work on a project has important connections with the research that she is doing for her thesis or dissertation, it may make sense to use her on grant-funded work. However, if employment on the project amounts to little more than a work-study job, the cost of training and supervising these students, may not justify the return.

Not all missteps had to do with personnel, however. The choice to use Flash for AXE's video tagging component is one I am not altogether sure I would make again. A closed source, high-level language such as ActionScript always has the potential to surprise the developer with nasty and unexpected limitations, and, indeed, about six months into the development process we were confronted with just such a surprise. ActionScript 2.0, it seems, adjusts for differences in network latency and processor speed by dropping frames during movie playback. This meant that it was unexpectedly difficult to perform actions such as jumping forward to a particular frame based on a user event with anything resembling precision. Various workarounds proved only marginally successful. Adobe has since released new versions of Flash and ActionScript, so it is possible that the problem has been solved in the current version. However, the lesson, I believe, is that the time one might initially seem to save by using a commercial development language with pre-built libraries apparently tailor-made for one's work may very well not be worth the cost of struggling with closed APIs with little ability to modify what lies beneath.

That is not to say, however, that using prebuilt libraries is never a good idea. At the outset of the project I performed an inventory of the various JavaScript/AJAX frameworks (at the time Prototype, Dojo, and Google Widget Toolkit were the most prominent) and was unconvinced any were documented well enough to warrant the time it would take to learn them (our team was fairly well acquainted with JavaScript programming already). About midway through the project, though, documentation for the various frameworks became better and cross-browser headaches made their use seem more attractive. In about the ninth month of the project I therefore made the risky decision to recode the entire project using the Yahoo User Interface (YUI) framework. The choice, I believe, was well made. The framework is extremely flexible and allows as much or as little native JavaScript as one desires to include. The syntax is intuitive and the code loads quickly and runs efficiently. Many of our cross-browser problems were immediately solved, and the code-review necessary to convert to the framework helped our team locate and eliminate bugs which had plagued our work for months.

There are now many more, well-documented JavaScript frameworks available, and I would encourage any PI beginning a new project to make a thorough review of each before beginning any development work. Recent releases of JQuery and Dojo would, I believe, make either a contender for a new project. The Google Widget Toolkit is also interesting, but I am unconvinced that the inevitable inefficiencies of compiling Java to JavaScript and the unreadability of the compiled JavaScript code would justify the GWT for any project with a team of capable JavaScript programmers (though for teams with a great deal of Java experience and relatively little web programming resources it may make sense). I have never regretted the choice of YUI, however, and would recommend the framework without reservation for those wanting to write cross-browser, object-oriented JavaScript.

Future plans

The beta release of AXE has some significant limitations. At present, the video tagger only accepts FLV movies and cannot load previously created tag sets for additional editing. The code library, while significantly more modular and portable since our migration to YUI, remains somewhat idiosyncratic and would benefit from a significant code revision and better documentation. Automation of version control (currently a far more manual process than we would like) along with automated “wizards” for tagging certain common sorts of regions (words in a page image, for instance) are also functionalities we expect users will desire. The aforementioned pending Mellon proposal will fund the development of the first two issues, and we have built automation into a Preservation and Access grant currently under consideration at the NEH.

Conclusion

Although AXE has only recently been publicly available, digital humanities scholars are already beginning to use our tool suite and seem pleased with what they have found. Future development will improve the tool, and its forthcoming integration into Zotero will very possibly make it the most ubiquitous multimedia annotation library on the web today. Although certain missteps (such as underestimating development time and the problems we would encounter using Adobe Flash) slowed our development, they did not, ultimately prevent us from meeting our goals and exceeding our expectations.