Citation:

Dickie, Grant, Jennifer Guiliano, Trevor Muñoz, Jim Smith, and Travis Brown. "ANGLES: A Web-Based XML Editor" National Endowment for the Humanities, Grant Submission, University of Maryland, College Park, MD, 2011.

Licensing:

Collaborating Sites:

University of Maryland
Maryland Institute for Technology in the Humanities

Team members:

Maryland Institute for Technology in the Humanities
Grant Dickie
Jennifer Guiliano
Trevor Muñoz
Jim Smith
Travis Brown
Kirsten Keister
Raffaele Vigilanti
Additional Contributors
Hugh Cayless
Doug Reside

Acknowledgments

Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the collaborating institutions or the National Endowment for the Humanities.

ANGLES: Level 2 Digital Humanities Start Up

Enhancing the humanities through innovation: Adding machine-processable information—"markup"—to texts, images, or other media is a core activity in many types of digital humanities research. Markup allows scholars to prepare their materials of study for computational analysis, publish these materials in a variety of formats, and record their interpretive judgements in standard form. To fully realize the benefits of working with digital materials in this way, humanities scholars need tools which specifically address their research needs but, at least as critically, humanities scholars need ways of engaging productively with developers and technologists.

The most likely avenue for the development of such tools is the open source software paradigm where "ownership" of technologies is broadly distributed among a community of users and responsibility for innovation and maintenance is undertaken by a distributed set of developers, many of whom are themselves also users. There are many examples of successful projects created using variations of this model, including the Text Encoding Initiative (TEI) standard, the Zotero citation management software, the Fedora Digital Repository software, and the Wordpress publishing platform. The vectors for humanists to engage in or influence the development of these projects are limited—the ability to write code and manage the mechanisms of distributed software collaboration (source code management, ticketing, and similar tools) are common requisites.

As a result, the design phase of tool development in the digital humanities is often either a small group endeavor or a one-to-one exchange where a humanist poses a set of potential research questions and desired functions, a computer scientist or team of developers build the technology, and the humanist returns to provide feedback and then begin using the tool. Driven by a small group of users' needs, many digital humanities technologies never advance beyond their initial concept—not because the idea was ill-conceived, but because without significant community buy-in, the technology does not fully engage with its full potential user community. Tools are often useful for an individual or a single project but not for the wider community. There should be more points of connection and exchange between the intellectual and methodological activities of humanities scholarship and the development process of software tools intended to serve the digital humanities community. Often, the technical revision process takes into account extensive feedback from user communities only after the technology has been launched and it can be difficult to keep potential users engaged between "releases" of a tool.

We propose a solution to the adoption gap that has grown up between scholars with digital materials and developers of more effective means of conducting research with those materials. By combining the model of intensive code development (a.k.a. the "code sprint") with testing and feedback by domain experts gathered at nationally recognized disciplinary conferences, we will create a tightly-integrated community-development loop that will allow community buy-in as the tool is being developed. This strategy was used very effectively by UVA for the development of the Hydra digital archives package. To structure interactions between users and coders at these events we will use a test-driven development methods which asks attendees to create "stories" to describe their requirements for the software. This creates "pseudo-code" which the developers can interact with. The project managers will ensure that community input is aligned with the larger development goals and timelines of the project. As a pilot for this "community roadshow" approach to tool development in the humanities, we

will develop a web-based editor for working with XML markup through engagement with the large and active community of scholars, teachers and developers who work with the TEI.

A web-based editor would serve the humanities by providing a lightweight, platform-independent solution that would not require software downloading. Instructors and/or faculty providing XML training would therefore not have to coordinate with IT departments or arrange for the purchase of expensive commercial software. This would dramatically reduce the current resource burden associated with experimenting with XML not just for students but also for humanities faculty interested in leveraging XML for their personal use, regardless of the scale of the project. Additionally, a web-based editor could be more easily integrated into other digital humanities projects since it offers an easy web-based function to markup digital texts. A broad example of this would be digital humanities projects utilizing "crowdsourcing", the process of asking general users to contribute markup. Having a common web portal for markup would boost buy-in for those kinds of efforts. Even robust digital projects like the Women's Writers Project at Brown University, the Walt Whitman Archive at the University of Nebraska, and the Music Theatre Online Project at the New York Public Library would potentially be served by this free, web-based XML editor.

Large, successful open source software projects have incorporated techniques such as identifying developer advocates and community managers to coordinate distributed development, encourage conformance with best practices, and provide points of entry for new developers throughout the entire development life-cycle. Zotero, a bibliographic management software package developed by the Roy Rosenzweig Center for History and New Media at George Mason University and Android, the mobile operating system developed by Google, are two notable examples of this process. Combining "code sprints," tightly-integrated feedback from domain experts, and developer relations—best practices from computer science and open source software engineering—will effectively address the need to generate users for technologies, provide better technical development by potentially having on-demand users to explore code as it is being created, and allow these communities to more clearly elucidate their intellectual, methodological, and technological needs.

Environmental scan: The field of XML editors is largely dominated by XML desktop-based editors including <oXygen/>, Eclipse, and XML Pro. Robust, these tools are quite useful for large-scale projects or major research universities who can invest the financial resources (in the case of <oXygen/>) and technical resources (in the case of Eclipse and XML Pro.) These tools rarely serve individual scholars or teachers from lower-resource universities and/or the general public. Currently, there are no web-based platform-independent editing tools that treat XML as anything other than plain text.

There are a limited number of platform-dependent tools that leverage XML: 1) The Son of Suda On Line (SoSol) is a web-based, fully audited, version-controlled editing environment being built for the papyrological community using Ruby on Rails (RoR), a programming environment for the Ruby code language. RoR depends on server-side scripts communicating with a centralized database which privileges using SoSol for ancient Greek and Arabic texts. As a result, SoSol restricts itself to a limited range of text markup which then impacts the ability to use the tool online. Users are almost required to use the tool offline which requires an advanced set up procedure in order to accommodate markup associated with prose, printed text, and English manuscripts; 2) The Islandora digital repository software (based on the Drupal content

management system and the Fedora Digital Repository) is currently developing a XML-TEI editing module. In order to implement Islandora software, users must install and configure a Drupal application, then the Fedora Digital Repository, and finally add on the TEI editing module. Users' installation of Drupal can use up memory and resources, making the client browser work slower in addition to a commitment to work within the Drupal and Fedora frameworks; and 3) "Ace: Ajax.org Cloud9 IDE" is a web-based code editor integrated with Github, the popular code management and sharing site. Ace attempts to emulate the environments of lightweight text editors, such as VIM, EMACS, and TextMate. As a general purpose programmer's editor, this tool lacks good support for XML including features such as code/tag completion, schema integration, and xml-aware mechanisms for tracking changes.

Intensive code development for computational tools, while uncommon in the digital humanities community, is frequently used within standard computer science development processes. It allows groups of code developers to rapidly develop a tool (or tool extension) with clear parameters for success within a dedicated and focused development environment. The adaption of this model to digital humanities has been quite successful: the CollateX team utilized a code sprint to contribute to a suite of RESTful services for text collation (2009) and the Center for History and New Media at George Mason University developed Anthologize, a WordPress plugin (2010). We intend to leverage humanists' expertise throughout the complete process and not just at selected moments by attaching the code sprint to professional conferences where humanists can provide immediate feedback as the code is completed.

History and duration of the project: ANGLES, constructed during a three-day intensive coding session, was prototyped by Jon Deering of St. Louis University, Hugh Cayless of New York University, and Doug Reside of the New York Public Library. The prototype coding team deployed a commonly-utilized coding process (the code sprint) to rapidly design, develop, and deploy the ANGLES demonstrator supported by remote collaboration tools including Skype and Internet Relay Chat. Working with code sprint attendees including staff at MITH, in its' prototype form, ANGLES provides a simple web-based tool for XML editing that is responsive to the challenges and needs associated with interoperable open-source XML tool usage. It currently provides limited syntax coloring and basic tag completion for a single XML schema. . With this prototype as its basis, the prototype coding team sought additional support from MITH that has resulted in this Level 2 start up application. Our development will be focused on expanding these tools to include customized project schema as well as improving its syntax coloring and tag abilities.

Joining the team for the 1.0 release will be J. Grant Dickie, Trevor Munoz, Jennifer Guiliano, and Jim Smith. Cumulatively, they provide additional expertise in Digital Data Curation, XML-TEI standards, Digital Humanities, and Computational Design that bolsters the already considerable expertise of Deering, Cayless, and Reside. With these additions, the ANGLES 1.0 development team anticipates subsequent phases of this project including: integration with Project Bamboo, a Mellon Foundation-funded project that will provide shared applications and infrastructure for humanities research environments and the integration of Encoded Archival Description schema for archival purposes. As each of these are open-source based community-backed endeavors, we believe that ANGLES 1.0 will potentially see further development from the coding communities embedded in these disciplines including, but not limited to: digital libraries, digital humanities, digital archives, and computer science. The cross-

pollination of our approach of attaching code sprints to professional conferences where specialists in these disciplines will be available for beta-testing, refinement, and feedback is clear. We anticipate these communities taking ownership of the various versions of ANGLES that will be generated leading to our 1.0 release, available as open-source software in Github.

Work plan: An open-source integrated development environment (IDE) for editing XML will be developed during the course of three pre-conference code sprints. This IDE will be built upon the work already done by Ajax.org Code Editor (ACE). ACE is an open source, extensible environment built in Python and Javascript programming languages and modular in its architecture. Light-weight in its resource needs and based on the client-side, developers for ACE can expand on the functionality to allow for more writing methods, format and markup support, and integrated tools support. Code sprints will be undertaken by a five-person team composed of Reside, Cayless, Deering, Dickie, and Smith in the two days preceding each conference, where agreed upon features and development "tickets" will be completed. Each sprint will consist of two days of pre-conference coding followed by a day of on-demand coding working with humanists attending the conference. One member of the coding team will be selected to attend each conference and provide on-site integration with developers located remotely. This development will follow a development schedule as outlined below. In brief, this roadmap includes an initial code sprint that will focus on development of IDE “basics”, i.e. schema validation, code auto-complete, code coloring and formatting, and backing up data with git. The two remaining code sprints will focus on concerns and suggestions from users, and on development of more customized schema for validation. Please consult the Appendix for an itemized list of activities by conference/code event. Via the chosen developers and social media streams, the complete development team would be actively engaging with scholars as they explore the proposed schema, receiving feedback related to modifications, and coding via three main avenues: 1) through on-site demonstrations (likely during poster sessions or in a book exhibit); 2) through twittering notes related to code modifications and request community feedback; and 3) via potential cross-pollination by requesting assessment by humanists from XML projects attending the event in appointment-style meetings. To aid the development team in managing these interactions and capturing the appropriate social media interactions, a developer manager will be present to facilitate development meetings and assure that features gathered through direct interfacing with conference attendees and Twitter are addressed, analyzed, and made into goals for development. The role of development manager will be shared between Trevor Munoz and Jennifer Guiliano. In this way, the milestones for the project will be acquired directly from crowd-sourcing from users within the field of Digital Humanities and then narrowed down and set into specific coding goals by the lead developer and development manager. We anticipate that we will be able to illustrate the importance of robust integration with tight feedback loops between humanists and computer scientists en masse through this live sprint-conference model.

Development, though, will not cease between these events. The attending developer will collect community feedback and present it to the full development team. While all developers will be required to read and analyze comments from attendees, it will be the job of the lead developer with assistance from the development manager to assess each comment and filter them down to three categories. These categories are: 1.) Necessary Additions or Bug Fixes. These are defined as items that must be included or fixed in the IDE open source code or else

the resulting platform will be utterly useless to the community. This category is reserved for "show-stopper" bugs that either crash or prevent users from completing tasks and functionality that is either already present in similar tools or has a wide-spread demand among the community (i.e. more than a few attendees asked for this feature); 2.) Possible Additions and Quirks. Reserved for those features for which attendees report fewer demands and bugs that can be classified as hard to re-create or are specific anomalies. These items will be solved only if level-1 additions and bugs are already completed; and 3.) Not considered. Reserved only for functionality requests that are either too large to complete in the time allotted the grant or are simply unrealistic. They will not be dealt with. Following this assessment process, these items will be posted under the heading of the conference as disciplinary feedback with the in-progress code being annotated similarly.

A web site will be hosted by MITH that will act as a record of the proceedings of sprints and the comments and interactions from conference attendees. As mentioned, code sprints are difficult to document as they happen quickly and often change directions and goals during the course of a single sprint. Direct recording of such events therefore becomes unmanageable. To accommodate the changing schedule, Github blame records, tickets, and commits will be recorded on the Github repository location and a link provided to these records on the MITH website. These resources will serve as records for posterity as well as ongoing and dynamic records of development for the development team manager to review. After each code sprint, a Github tag, or snapshot of the code, will be set up on the main Github repository. Outside developers and coders will have the choice of “forking the code” (a.k.a. utilizing a segment of code for their own differing track of development) at significant points of development. Reviewers can also view the code and see its progression through the different stages.

Often during start up development projects, the developers are separated from the rest of the test users. This project seeks to experiment with avoiding this facet of the development process. Each developer and manager of the team will be encouraged to have links to their own blogs and incorporate contributions to the main blog on the project website. Biographies and pictures of each of the team members will be presented on a page of the website, with lists of their blog contributions and Github commits and pushes underneath the respective team member. Conference attendees can watch and keep track of development and read blog entries on the website, making the experience of development more interactive and personal. The website can even be designed in a more entertaining and brighter fashion, similar to sites like Github and Twitter. This is done to ensure test users feel comfortable submitting suggestions, bugs, and general feedback to the developers. Likewise, developers will be encouraged to learn from scholars, the research trends in the field as presented in conferences, and each other.

Evaluation of each code sprint will occur after the conclusion of the code sprint and the following conference. Evaluations will focus on what changes were made to the central repository and what milestones were added, removed, or altered during the course of the interactions with conference attendees. The development manager will then evaluate the performance of each coder and go over goals for the team. A larger, all-encompassing evaluation will be conducted following the conclusion of all three code sprints. This will assess the project as a whole and determine next steps for the project. These next steps need to address how the code developed and distributed on Github will continue beyond the code sprints. In addition, documentation will have to be created during the final evaluation for users to

implement the software. This documentation must cover installation of the tool and proper usage of the tool, as well as advanced documentation for developing plugin modules. This documentation will be available on the project website, included with the software, as well as code-specific comments within the source code itself.

Staff: ANGLES will be administered by Project PI, J. Grant Dickie, web programmer at MITH. The project team will include: Hugh Cayless (NYU), Jon Deering (St. Louis University), Doug Reside (NYPL), Jennifer Guiliano (MITH), Trevor Munoz (MITH), and Jim Smith (MITH). Additionally, as our project will rely on live on-demand open-source development, we will issue a call for participants on our website and preceding each conference so that additional developers may join the team. They will be selected and managed by the PI.

Final product and dissemination: Our project will release all of its work as open-source code thereby encouraging other researchers to use our work, benefit from our investment of resources, and alter our code to extend the efficacy of ANGLES 1.0. We anticipate on-site code sprints and the associated refinements at the 128th Annual Conference of the Modern Language Association (3 to 6 January 2013: Boston, Massachussetts), the Alliance for Digital Humanities Organizations' Annual Digital Humanities Conference (Spring 2013: Lincoln, Nebraska), and Code4Lib 2013 (TBD).

Additionally, we will release a reflective white paper at the end of the grant detailing the challenges and successes of experimenting with the integration of code development via the sprint model embedded within professional conferences. As part of this white paper, we will also gather and archive user interactions with our development team at these events and throughout the development phase via social media apparatus including Twitter and Blogs. These will be archived on our project website and will form a useful thread throughout our final reporting efforts as we attempt to extrapolate potential understanding of how the library, digital humanities, and literature communities wish to interact with XML. These extrapolations will form a core set of recommended best practices for other developers wishing to work in interoperable tool design within these communities.