

MOAC: Community Toolbox Project Technical Analysis

Patrick L. Schmitz
Ludicrum Enterprises

INTRODUCTION

The MOAC: Community Toolbox project set out to develop a software tool that enables production and sharing of standards-based content, and to share this tool with the cultural community through a 'community toolbox' site. Building on previous successful work in this area, the new MOAC tool was developed as both a utilitarian tool and as a test case for exploring more general issues of content sharing and community tool development.

The MOAC: Community Toolbox project was funded through a grant from the Institute of Museum and Library Services, and completed the development phase at the beginning of 2007. As part of the grant, a final report is generally produced for the funders and the broader community. However, rather than just a summary report, it was proposed that a more thorough technical and logistical analysis be undertaken of the MOAC: Community Toolbox results, to the end of producing both a summary report of the work completed as well as recommendations for improving the implementation.

We briefly summarize the context for the MOAC: Community Toolbox project, including the existing standards and practices with respect to metadata schemas, related work on metadata translation, and the technology context (both what is commonly in use in the community, and what was available for development of the Toolbox). We consider the expressivity/fidelity of the output XML, and the choices made in this context. We then present an analysis of the implementation including both usability as well as efficiency/scalability perspectives. We consider possible alternatives to the existing application model and the potential for transitioning the current project to these new models.

Finally, the MOAC: Community Toolbox team wished to make the associated software and related materials available to the community as an Open Source project. However, the project team is not familiar with the requirements and principles of the Open Source development community, nor with the various license options that may be used. This is a not uncommon problem, and so we include some background and recommendations for making a successful transition to an Open Source development model.

BACKGROUND AND CONTEXT

The project proposal described the interest in supporting learning and research through the sharing of digital resources among cultural institutions. It also described the significant challenges faced by museums trying to expose their collections, noting that "each museum or library usually participates in only one such project because of the cost, labor, and technical complexity involved....For most museums and libraries, producing content in standards-based formats, such as Encoded Archival Description (EAD [7]), Metadata Encoding and Transmission Standard (METS [11]), or even Dublin Core (DC [6]) in the proper Extensible Markup Language (XML [9]) format, is enormously complicated and beyond their technical capacity."

Most metadata standards of interest have an XML format, which allows (at least in principle) for the translation between formats. The basic approach is therefore one of translating from an existing metadata format in some simple form (such as a comma separated values (CSV) listing) to a central XML schema and then translating this to the various output schemas desired (METS et al.).

There have been previous efforts such as the 7train project from CDL [1] to provide XSLT conversions between METS and OAI. The OAI Metadata Harvesting Project (Prom and Habing [16]) explored the challenges of transforming standards for cultural heritage archives such as EAD to OAI. More recently, groups such as The Getty have begun to develop cross-walks (an analysis of how to translate from one schema to another) among various archive-related XML standards (see note for [2]). However, these prior efforts do not provide an end-user application appropriate to the intended audience.

There was considerable discussion and concern that the MOAC tool be lightweight to install, and that it leverage technology that was already familiar to museum staff (especially at small to medium-sized institutions). There was general consensus that the most commonly used database system among these institutions was FileMaker Pro (FMP) - most museums use FMP in some capacity and have staff familiar with its use. Although the core of the tool would involve XML translation via XSLT, most museum staff would be unfamiliar with XML processing tools, and would not be comfortable configuring these in their environments. Various programming frameworks (e.g., Python, Java) wrap these tools and would support application development, but these would require the end user to install and configure the framework, or would require the project team to develop a complete installer. This would have required considerable resources and would have yielded a fairly large installer package, both of which argued for the simpler FMP approach. FMP supports CSV file input, a UI for associating the CSV fields to a schema, and has a built-in XML/XSLT processing engine, and so seemed to be the most promising framework within which to develop the MOAC Community Toolbox. Nevertheless, the decision to build upon FMP had some significant implications for the resulting tool, as discussed below.

THE METADATA TRANSLATION MODEL

A primary point of evaluation is the means by which metadata is translated from various input schemas into the desired output schemas, and the *fidelity* of this translation - the extent to which concepts in the original schema are reasonably expressed in the output schema. This was naturally a significant concern of the design team, and they set about analyzing the problem in a fairly methodical manner. A field-by-field equivalence table was developed that considered the schemas of each partner in the project, and several others as well. Each partner initially gathered the fields from their respective schema that the partner considered important to expose to the aggregators. This would determine how the respective collections would be modeled, and how they could be searched and explored.

All of the partners were versed in the expression and application of metadata for collections, and two of the partners in particular (RLG and CDL) have extensive experience with metadata expression, standards and application. After some consideration of alternatives, CDWA-Lite emerged as the favored candidate for a central schema. Although the XML schema for CDWA was relatively new, it provided the best coverage with respect to each of the other vocabularies. The analysis of alternatives seems quite solid and the choices well justified. The metadata provided by the partner museums, including those in a popular commercial content management system (EmbARK [8]), were covered by the CDWA-Lite vocabulary, and so the schema-level

translation seems to be of high fidelity.

It is worth noting that some of the target standards such as METS do not explicitly require detailed descriptive metadata for objects in a collection, although they do *allow* it. Some aggregators recognize that this richer metadata is necessary to support more effective search and discovery by end-users. Some encourage the inclusion of richer descriptive metadata, and at least one (CDL) requires richer metadata before they will aggregate a collection. The goals of the MOAC Community Toolbox project were to support this rich level of metadata more easily by converting whatever a museum had to the standard formats.

The next step is then to consider the semantic fidelity of the translation. That is, to what extent is the context and meaning of the original metadata maintained and does it provide the desired utility to users of the aggregation services: can end-users reliably discover and locate collections and objects exposed by participating organizations. An implicit but important assumption here is that the exporting institution *wants* to expose their collection, and to make the objects in the collection easy to find. This is something of a departure for many museums; unlike libraries, museums have not always recognized a mandate to make their collections easily searchable. Many collections managers have yet to come to grips with the implications of exposing their "treasures" in this manner, and very few have embraced the idea that the exposed metadata and the associated search and discovery tools should make it as easy as possible for any member of the public to explore the complete collection behind the museum. In practice many museums expose only a portion of the collection, and the intended audience for each collection manager may vary from other museums to researchers and perhaps the general public. This in turn has an impact on the choice of metadata that they choose to expose, and the desired effect of exposing it.

The end result of all this is that it is quite complex to evaluate the semantic fidelity of metadata translation; such an evaluation requires a concomitant review of the context for each museum. For each case, different success metrics might be appropriate:

- Were other institutions able to easily find items in a given collection? Did the metadata exposed match the vocabulary common in the respective domains?
- Was sufficient metadata available for researchers to do their work using aggregation sites, supplanting the need to work with individual institutions?
- Did public traffic to the aggregation websites increase with the addition of richer collection metadata? Were there more links into the aggregation websites, or to the partner institutions, after the collection metadata was exposed?

It will take time to observe these effects, and to correlate them to the goals of the respective institutions. An additional line of research beyond the MOAC Community Toolbox project might also consider the extent to which the goals of the institution in exposing their collection impacted the richness of the metadata exposed, and whether specific patterns emerge. At the same time, an evaluation could be performed on the effectiveness of different fields, vocabularies and other aspects of the search and discovery mechanism. This would be compared to varying audiences (end-users of the aggregation services), with the goals of finding patterns that would guide the entire process.

There are a number of projects exploring new techniques for search and discovery of museum collections, ranging from end-user tagging of objects (as in the *steve.museum* project [17]) to projects leveraging other

social media techniques for online museum collections (e.g., Delphi [5]). It is safe to say that these exploratory projects are adding additional facets to the metadata associated to a collection, notably the *response* of users to objects, their interest level and the associations of objects to user-generated narratives. It remains to be seen how the metadata created by and for end-users will differ from that generated by traditional curation processes, and how or even whether any of this new metadata will be integrated into the traditional workflows within museums.

USABILITY AND RELATED ANALYSES

As with any software application, it is important that the tool be easy to learn, to understand and to use. The analysis of this is now a domain unto itself generally referred to as Human Computer Interaction, and more specifically within that domain the field of Usability Analysis. Many of the recommendations of usability analysts center around the use of standard techniques for typical interaction use-cases, so that users accustomed to common paradigms will be on familiar ground. In addition, there is a general consensus that applications should support easy *discovery*, usually by making the UI for common operations clearly visible, following typical naming schemes, and supporting embedded documentation such as tool-tip text, status feedback, and "wizards" or dialog flows that guide users step by step through more complex tasks.

The designers of the MOAC Community Toolbox are experienced FileMaker Pro developers, and not unfamiliar with principles of usability. For the most part, the implementation reflects this, following good usability practices, however the developers were constrained in some respects by the limitations inherent in the FMP environment, which is not really intended as a rich application framework (significant effort was required to get the Toolkit to look as nice as it does). The Toolkit uses forms and labeled buttons to guide the user through common functions. Where possible, dialogs inform the user of next steps, or at least of problems when the intended sequence is not followed. In addition, a "quick start" user manual is being prepared that assist users in learning the tool and being productive with it. Most of the core functions are supported in a direct and relatively easy to understand manner, and the user interface has already benefited from internal testing and an external review that included usability issues. Nevertheless, there are some rough areas and the lack of certain capabilities in the FMP environment led to some usability problems, which we briefly describe below. Because these are problems inherent in the FMP platform, it will be a challenge to remedy them.

There are a number of usability issues associated with loading and saving files to and from the local filesystem. The FMP controls do not allow application software sufficient control over the standard dialogs to set defaults and other sorts of operations that can improve usability. When generating HTML output, the tool cannot create a new output folder and so the workflow is somewhat confusing and inconsistent with most applications. There are some third-party plug-in extensions that provide workarounds or improved functionality for at least some of these problems, but using these would require the end-users to install these plug-ins. This was deemed to be problematic for the target audience and so was rejected as an alternative.

One fairly significant problem is the lack of a reasonable way to interrupt long operations, such as the export of a large collection (more than a few dozen items). The FMP toolset does not allow applications to support a cancel operation that can cleanly cancel a long operation. Because export operations can take considerable time, the lack of a cancel is a serious usability problem. There seems to be no work-around for this at this time.

There are a number of personalization features that could be improved that would make the tool easier to use. By saving values such as the default load and save locations, and default Base URL values for generating output, users would not have to type these in each time. This was identified in an external review of the tool, and may be addressed at a later date as part of an ongoing maintenance process.

There are also a number of minor issues related to things like tool-tips, context sensitive help and other smaller user interface issues. While these have some impact on usability, the greater impact is probably on the general "fit and finish" impression. It is not unreasonable to expect that the museum users will overlook these blemishes, at least in an initial version of the Toolkit.

As with any application, continued use and review by a larger audience will likely reveal other issues that offer room for improvement, however these can be dealt with in an ongoing review and maintenance process.

PLATFORM DEPENDENCE, AUDIENCE FAMILIARITY

In addition to usability with the interface, there are a number of utility criteria that are important to consider. These cover points such the suitability of the application framework to the end-users' existing environment, and the workflow models supported by the application. In this respect, the MOAC Community Toolbox does quite well. The FileMaker Pro environment is available on a range of operating system platforms, and is used in most institutions that are candidate users of the toolbox. In addition, many of these same institutions have staff who have some technical facility with the FMP, supporting the possibility of community involvement in the ongoing maintenance and improvement of the toolbox (see also the section on Open Source issues, below).

The workflow for users should be fairly straightforward, and the needs of many users will be met directly and efficiently by the Toolbox.

The MOAC Community Toolbox does require the latest version of FileMaker Pro to run, and so some institutions may have upgrade their systems in order to support the Toolbox. However, this is not a significant cost (a upgrade is only about \$135 after an educational discount).

PERFORMANCE AND SCALABILITY ISSUES

Much of the functionality of the Toolbox is implemented in such a manner as to be responsive and to consume minimal compute and storage resources. However, there is one aspect of performance that has an impact both on the perceived usability of the tool, as well as on the ability of the tool to scale in application to very large collections, or to perform repeated updates of moderately-sized to large collections. Once again, these are not so much a matter of inefficient design or implementation models as they are of constraints in the FileMaker Pro database implementation. A number of operations generate output by first gathering together information from several tables within the database and then generating XML output from the resulting merge (a.k.a. a table "join" in database parlance). The obvious (and also the most efficient) manner to implement the output process would be to generate the XML for the entire merged list, and then to transform the resulting XML using stylesheets (in XSLT) to produce the final desired format. Unfortunately, in FMP, there is no way to perform such a merge and then generate XML for all entries in the resulting merged list. As a result, each entry is merged separately (joined into a temporary table), output as XML and then transformed and saved to an output

file. This makes the process quite a bit slower than would be expected.

In addition, there are serious performance problems with the built-in XSLT engine that initially made FileMaker Pro a good choice as a framework. It seems to be inefficient and not well optimized, and so the conversion process from one XML schema to another is (again) much slower than would otherwise be expected. While the target users may not be expert in the domain of XML and XSLT, they will likely have some comparable basis for comparison and are likely to note the slow processing speed. On the other hand, since the users have no comparable tool for the tasks at hand, many will just accept the performance so long as they can get the job done.

The performance can be both a usability problem as well as a problem for scalability. As noted above, users cannot cancel out of a very long output process, and the tool is effectively locked up while it is processing a large collection. In addition, however, the performance of the tool means that for large collections, the output generation speed will be a serious issue that will constrain certain applications. No careful benchmarks were performed, however some general observations were made. On a relatively modern personal computer, the processing took several seconds per output record. This corresponds very roughly to a rate of about 1000 records per hour. For collections under a few thousand items, this is reasonable enough and should not cause any serious problems. However for institutions exposing very large collections, this may be untenable. Certain museums (e.g., in anthropology, the natural or physical sciences), have collections numbering in the hundreds of thousands of records. To process such a collection would require several days or even *weeks*. It would also require that the process never be disturbed since the tools cannot resume a partially completed process. While a great many of the museums among the intended audience have collections small enough to easily fit within the performance constraints, there may be at some who will face challenges using the tool in its current incarnation.

MAINTAINABILITY AND GENERAL DESIGN ISSUES

No application is complete when it is released. It must be maintained to address unforeseen problems (bug fixing) and to support changes in the external requirements (e.g., changes to metadata standards or to the fields and content required by aggregation portals). Much of the ease of maintenance is determined by several factors, which we describe and then consider in this context:

- *Will the original development team be maintaining the tool? This is in some ways ideal since they should be familiar with their own design and implementation. However over time, most software developers will begin to forget even their own designs and will struggle to make changes and improvements without considerable effort, and so while this can be a mitigating factor, it is not sufficient to ensure maintainability.*

In the case of the MOAC Community Toolbox, it is not at all clear who will be maintaining the project. In fact, as the stated goal is to turn the project over to the larger community as an Open Source project, the developer team will necessarily come and go. In the near term the contract developers may be available, but this factor may be an issue for the Toolbox.

- *Does the design and implementation follow best practices of modularity, integrated test support and careful commenting? These are the "good housekeeping" factors in software development that are too often overlooked and that as a result can have a serious impact on cost to maintain software. It is not*

uncommon in the face of poorly modularized and poorly documented software (sometimes referred to as "spaghetti code") to reject many upgrades as too risky, and/or eventually to scrap the code altogether and start over (and the choice is not one of exasperation, but of simple expediency: it is actually cheaper to start over than to completely understand the existing system).

The developers made an effort to decompose the script methods and to comment them well. In addition, the core transformation logic is contained in XSLT that can be separately maintained. Nevertheless, the FMP environment placed certain constraints on the project that impacted modularity. Much of the script code is tied to forms and input buttons that are closely linked to the FMP framework. Certain aspects of the code (e.g., the XSLT stylesheets) are not stored in obvious places (e.g. as an external file) and so are harder to find and edit. This was a conscious decision to make distribution and installation of the Toolbox as simple as possible (the lack of external files simplifies configuration in a range of environments), however the flip side of this will be some increased maintenance costs. So long as the Toolbox remains in the FMP environment, the problem should not be severe, and someone experienced with the code could abstract much of the core functionality with moderate effort.

- *Does the implementation use an environment that is standard and widely used in the field? If obscure programming languages are used or if platform requirements are non-standard, it is much more difficult to find developers familiar with the environment, and it may be more difficult to keep current the application and platform interdependencies.*

The FileMaker Pro tools are very widely used among the intended audience for the MOAC Community Toolbox, and there are many people who are familiar with the basics of design and implementation in the FMP environment. However, among the broader community of software developers, it is not nearly as common as other database engines, and is not commonly used as an application platform. There *is* a reasonable pool of experienced FMP developers and so this is by no means severe, but it may have a negative impact on the longer-term maintenance costs for the project.

The issues and our conclusions beg the question of what alternatives might have been considered and how they might have fared. Most typical platform development tools would have a host of portability issues that would make them inappropriate choices for an environment that has a broad mix of computers and operating systems in common use. However there are some cross-platform solutions worth considering for the sake of discussion:

- **A Java-based application** would have been much more flexible and a more productive environment to develop interactive UI, provided the team had access to experienced Java developers. Although Java applications are not always as clean and visually pleasing as platform specific applications can be, the Java UI would be at least as nice as that afforded by the FMP tools. The performance would have been significantly better, including very fast XML and XSLT processing (there are ways of controlling the XML processing pipeline to make the process very efficient). Any of a number of database solutions could have been integrated, although some (like MySQL) would have required a somewhat larger and more complex installation package (MySQL installers range from 40 to 60 MBytes, or about one-third to one-half the size of the FMP installer required for any users who have to upgrade to the latest version). Java applications can integrate with FileMaker Pro, and so could use the same back end as the MOAC Community Toolbox did, but would then still be subject to certain of the performance problems. Java is available and generally installed on most personal computers, although runtime versions can be an issue.

Some care could be taken not to develop the application in such a manner as to require an upgrade. A complete upgrade and installation package is on the same order of magnitude as an FMP upgrade installer, so this is not much of a differentiator. All in all, Java might have been a viable option, providing much better performance but with more installation weight and complexity.

- **A Flash-based application** would have been somewhat more flexible and a more productive environment to develop interactive UI (again, requiring access to experienced Flash developers). Flash applications are difficult to develop according to good engineering practices, but it is possible, and the resulting UI is generally quite pleasing. The performance would have been somewhat better, including reasonable XML and XSLT processing (while not highly optimized, Flash support for XML is much better than in FMP). Any of a number of database solutions could have been integrated, and so the same comments and requirements as for Java apply here as well. Flash is widely available, cross-platform and a fairly up-to-date version is generally installed on most personal computers. There are a great many visual designers with Flash experience, but many fewer solid software developers working in Flash (although this is beginning to change, especially with the advent of ActionScript 3 and associated development tools). Flash could also have been a viable choice, providing nicer UI and better performance but with the same installation and configuration challenges as Java.

We describe these alternatives not to second guess the Toolbox project decisions, but rather as a starting point for others considering a similar project. The priority on a simple installation and configuration made the FMP platform a good choice for the MOAC Community Toolbox. If the potential use-cases include more large collections, then an alternative might make more sense. In addition to two platform alternatives above, we also consider a more radical departure in the next section that might address a number of the current issues and provide other advantages as well.

AN ALTERNATIVE MODEL

One of the biggest changes sweeping the industry in the phase sometimes characterized as "Web 2.0" is a reorientation from traditional application models to Service Oriented Architectures (SOA's). In many different domains, web services are being developed that provide traditional application functionality via the web browser, with any serious computing that may be required performed on the server. Examples range from online video editing and photo management to office document processing to tax preparation. In the SOA and rich web applications ("web-apps") model, there is little concern for platform issues other than the development of cross-browser functionality, and much of this is being abstracted for developers by packages such as JQuery ([10]). There is also no need for distribution concerns, since anyone can begin using the service simply by directing the browser of her choice to the web service URL. The services can be gratis or fee-based, and can be open or controlled with tight security. Because of the general enthusiasm for this model and the speed with which it is being adopted, there are a great many developers familiar with the development model and tools. All that is required is a hosted service with support for a basic database (e.g. MySQL) and code (PHP, Perl or Python). This can be had for a minimal cost (on the order of US\$10 to \$20 per month for varying levels of bandwidth and other services required).

We can consider the basic goals of the MOAC Community Toolbox as a set of basic services to be provided to the end users. These include:

1. input of metadata in some simple formats

2. some basic editing and manipulation of the metadata
3. conversion to output formats adhering to one of several metadata standards
4. the implicit if currently unsupported goal of publishing the resulting standard metadata to the aggregator/portals of the user's choice.

The first service just requires file upload and CSV parsing, for which there are many libraries available. A UI would have to be developed to map from columns in the CSV to the CDWL-Lite (or equivalent) schema, but this is relatively straightforward.

There are already a host of web interfaces to interrogate and update databases, and more specifically museum collections. This is fairly straightforward development, for which many productivity tools are available to speed the process. An initial version could provide simple update of the fields in CDWA-Lite, and later versions could expand this as users desired.

The conversion to standard metadata expressions would leverage the existing XSLT stylesheets and standard XML/XSLT processing engines, and so would require a very small amount of additional work over what has already been done. Because an advanced XSLT processor could be used, the performance would scale to much larger files, even as a shared service.

Finally, users could download the results and save them to a local file. In addition, if the user was content with the conversion and wanted to submit the results to an aggregator, the web service could easily support this with the click of a button, providing an end-to-end solution for the museum staff. The back-end process required to support this is trivial, given the commonly available tools and libraries of the web application environment.

While the approach would require some rich web UI coding, developers would work with a UI toolset that is at much richer than that provided in FMP, and that is available to everyone with a web browser. This model would also more easily allow for improvements and extensions to the tool, since there would be no need to redistribute to users - they would get the latest version whenever they came to the site.

Such an approach likely never even came up in the early discussions of the MOAC Community Toolbox, but it might well represent a possible alternative for a future version of the Toolbox. Considering all the aspects of development and maintenance, usability and utility, such a model offers advantages over the current FMP implementation, although some users might need to get used to such a model and to trust a central service like this. This approach can leverage much of the design logic and at least core pieces of the implementation, and so builds upon the current project. It could also provide advantages as the project is transitioned from a closed development model to an Open Source model, as we discuss in the next sections.

At the same time, such a model naturally requires someone to provide and support the infrastructure for such a centralized service, which is a real issue. It also may make it more difficult to customize the tools or the translations for individual applications or institutional needs. The approach needs deeper analysis of potential problems and the various issues must be considered against the advantages, but such an alternative approach seems worth exploring in a future version.

MOVING TO OPEN SOURCE

The title of the MOAC Community Toolbox project underscores the commitment to providing a solution for the

community, and the project structure incorporated the contributions of many partners in this community. It was the intent of the project team that the resulting tools be provided to the community under an open license, and that the Toolbox would be maintained by the community following the model of open source development. The Toolbox that has been created already represents a Public Good in the economic sense, and will provide an important seed for an ongoing project. However, there are a number of issues that must be understood to establish an open source project and to make it successful. This section will introduce licensing issues, the social dynamics of open source projects and the associated requirements to draw a community and allow them to be effective contributors.

There are many articles and several books that cover these issues in considerable depth. Steven Weber's "The Success of Open Source" [18] provides a good introduction and covers the licensing issues in some detail. In Osterloh and Frey [13], Osterloh and Rota [14] and other CREMA papers the authors provide detailed analysis of the social dynamics and "social economics" models that underlie the community creation of public goods, especially in the domain of software and information products. We draw upon these sources and others, but the original sources provide enlightening detail.

CHOICE OF LICENSE AND TERMS

Although the popular notion of open source is that such software is free and free to use, these are both misconceptions, or at best oversimplifications. Some open source software is sold, some is free. All open source software comes with a license and/or terms of use that bind the users and developers. An important aspect of open source development is the idea that the source code be available for all to see (i.e., "open"), and that those who extend an open source project give back to the community. There are variations on the license that provide more or less flexibility in this latter respect. One style of license allows anyone to extend a project as they wish, including for commercial purposes, and without requirement to donate any extensions back to the community (it just requires acknowledgement and notice that the original source is included or otherwise available). Some feel that this may spur adoption as it places fewer constraints on potential adopters. Another style of license that appeals to many open source purists requires that as part of the right to use open source software in development, any project that extends the base and distributes the results (whether for free or as a commercial product) must donate all extensions, bug fixes, etc. back to the community. The purists feel that this model is the only one that will support a sustainable open source community. There are many variations on these themes with subtle differences, the details of which are out of scope of this discussion.

The MOAC Community Toolbox team must decide what sort of license model fits their goals. Among the considerations are whether commercial vendors in this market are likely to adopt the work and incorporate it into their products (for sale). This might provide a broader distribution channel and so get the tools into more hands more quickly. However, if the vendors need not contribute bug fixes and extensions to the community, smaller institutions with tight budgets could be cut off from the latest versions. Moreover, the vendors might absorb or employ the talent pool who would otherwise contribute to the open source project for free. If the developers then lack the time or have the typical contractual obligations with respect to intellectual property, they will not support the project and it may well be unsustainable. Taking into consideration the modest commercial market for these tools and the modest resources required to maintain them, it should not be unreasonable to require a more restrictive license (e.g., one of the GPL variants) that ensures improvements are available to the broad community, while still allowing commercial vendors to leverage the work of the

community and distribute it widely as part of their offerings.

THE SOCIAL DYNAMICS OF OPEN SOURCE DEVELOPMENT

The idea that a major operating system or the services that power the web could be developed by a host of engineers spanning the globe with little formal structure and no monetary compensation strikes many as improbable if not impossible. And yet Linux has emerged as a significant market force, and Apache drives the majority of servers on the web. However, there are some subtle but important motivations and interactions going on within the community. Sociologists who study open source and other public goods creation have proposed a series of models to explain the motivations of the developers, including "gift economies", "reputation economics" and others. While it has become more common for open source developers to be employed to contribute (e.g., IBM pays many developers to maintain and extend Linux), most projects still survive on the free contributions of developers working in their private time.

In contrast to the high profile success stories like Linux and Apache, there are many many projects hosted at open source repositories like SourceForge (<http://sourceforge.net>) that are stagnant, with little or no activity and very few people contributing to the effort. The difference between a project that grows into a self-sustaining effort and one that stalls out must be understood if the MOAC Community Toolbox is to succeed as open source.

Coleman [4] describes the impact of what he calls "Rational Zealots", driving the creation of public goods. Basically, it takes an inner core of passionate contributors who are doing it not for the money, but because they believe passionately in the importance of building and sharing the software (or other productive results). These *intrinsically motivated* contributors usually bootstrap a project and provide the essential vision and momentum to drive the project forward. At the same time, Margit Osterloh and her colleagues argue that that the intrinsically motivated contributors are generally insufficient to sustain a project, especially if it attains any degree of success among users. Open source projects scale up when they are joined by *extrinsically motivated* contributors - i.e., those who hope to gain reputation, connections or even job offers as a result of their contributions. However these extrinsically motivated contributors have many choices of projects where they can devote their efforts, and they are judging which projects are most likely to provide them the desired return. Their choices usually rest on judgments about how high a profile a project has, how widely it is used, who the core contributors are, and whether they can offer sufficient value to be recognized for their efforts.

Thus to be successful, the MOAC team must identify a core team of people who will be the rational zealots for the Community Toolbox. There are several candidates among the project contributors but three or four should be prepared to work passionately on this. They will also be responsible for verifying the quality of work contributed by additional developers. These contributors might be supported by their institutions, so long as their involvement appears to be intrinsically motivated and not just because they are paid to do it. In addition, the core community of institutions can contribute by marketing the project and the tools to the broad range of potential users, generating momentum for the Toolbox and raising the profile of the associated open source effort.

An additional factor in this equation is the number of developers familiar with the product platform and the development environment (the talent pool upon which the project must draw). The more mainstream the tools and platform, the easier it is to draw in contributors. Choosing unusual or difficult to use tools will make it

harder to draw in extrinsically motivated developers. In the case of the MOAC Community Toolbox there may be engineers interested in contributing to a museum software project, but if they are not also experienced with FileMaker Pro development they may conclude that it will be hard to distinguish themselves on this project and so pass it over in favor of more promising opportunities. It remains to be seen if this is the case or not, but we see it as a potential difficulty.

A final factor that is essential to success in this realm is the structure of the project itself. The nature of open source development is distributed and disaggregated. Developers must be able to work on an isolated piece of the total project in a fairly independent manner. If the different pieces are too interdependent, the tasks will require increased overhead in the form of communication and coordination, and the project will suffer. Best practices in software design dictate a *modular* approach in which components are cleanly distinguished and abstracted, in part because it supports development with less overhead (open source or not). An initial task for the core development team will be to define components and ensure they are sufficiently separate and confined to support independent development. For example, the user interface associated each feature should not be tied too tightly to the logic that performs the associated function. The XSLT stylesheets that perform the schema translations must be independent of the database query model, etc.

There is good reason to believe that the MOAC Community Toolbox could be a successful open source project. There is a solid basis upon which to build, and a clearly identified need among the potential users. If a core team can be assembled, and if project partners visibly support the work, the outlook is good. In addition to good documentation of the existing work, it would make sense as part of preparing the transition to open source to gather up existing feature requests and bug reports in to an annotated list of potential tasks that contributors can take on. To be successful as open source requires some commitment, but moderate resource inputs have the potential to yield much greater results than would be otherwise be possible within the constrained budget of any individual institution.

CONCLUSION

The MOAC Community Toolbox project has achieved what it set out to, building a tool that will enable a broad range of museums and similar institutions to convert their existing collections metadata to standard formats suitable for harvesting by aggregators and portal sites. The team considered the needs and existing practices of their potential "customers" and chose a lightweight approach based upon the widely deployed FileMaker Pro environment, and industry standard tools like XSLT stylesheets. While there are some usability and performance/scaling issues with this platform, a high proportion of users will be well served by the resulting Toolbox. The project partners must now transition the tool to an open source model to sustain development beyond the end of this grant period. While they must attend to the requirements and social dynamics of the open source model, a good opportunity is there and justifies the investment in bootstrapping an open source project. One possible direction for an open source project would be to pursue the alternative, services oriented model for the Toolbox, perhaps as a "fork" in parallel to maintenance of the current tools. This approach would address some of the current issues, and might also draw in more contributors given the current interest in this kind of architecture and the desire of developers to enhance their skills and reputation in this kind of work. In any case, it will make sense to continue monitoring the deployment of the Toolbox, tracking uptake and noting problems and opportunities for improvement. Only with time will a clearer picture emerge of the real utility and success of the MOAC Community Toolbox.

REFERENCES

1. 7train project, supported by CDL and documented at <http://sourceforge.net/projects/seventrain> (last accessed on 25 June 2007).
2. CDWA Lite, Getty Research, documented at: http://www.getty.edu/research/conducting_research/standards/cdwa/cdwalite.html . An extensive crosswalk of CDWA Lite with other metadata standards is documented at: http://www.getty.edu/research/conducting_research/standards/intrometadata/metadata_element_sets.html (both links last accessed 25 June 2007)
3. CED (2006), "Open Standards, Open Source, and Open Innovation: Harnessing the Benefits of Openness, A Report by the Digital Connections Council of the Committee for Economic Development, April 2006
4. Coleman, J. (1987), Free Riders and Zealots. In K. Cook (Ed.), *Social Exchange Theory*, (pp 59-82). Newbury Park, CA: Sage Publications.
5. Delphi: An online museum collection browser (2007), Amuzinskaya, O., Hilgert, A., Lesser, J., Schmitz, P., Yu, G., U.C. Berkeley School of Information Masters Final Project report. Available at: <http://www.ischool.berkeley.edu/files/DelphiFinalReportLinked.pdf> (last accessed on 25 June 2007).
6. Dublin Core Metadata Initiative. Documented at <http://dublincore.org/> (last accessed on 25 June 2007).
7. EAD, documented at <http://www.loc.gov/ead/ead.html> (last accessed 25 June 2007).
8. EmbARK collection management system, Gallery Systems Inc. Documented at <http://www.gallerysystems.com/products/embark.html> (last accessed 25 June 2007).
9. Extensible Markup Language (XML). Documented at: <http://www.w3.org/XML/> (last accessed on 25 June 2007).
10. JQuery Javascript library. Documented at: <http://jquery.com/> (last accessed on 25 June 2007).
11. Metadata Encoding and Transmission Standard (METS), Documented at: <http://www.loc.gov/standards/mets/> (last accessed on 25 June 2007).
12. Open Archives Initiative (OAI). Documented at: <http://www.openarchives.org/> (last accessed on 25 June 2007).
13. Osterloh, M. & Frey, B. (2000). "Motivation, Knowledge Transfer, and Organizational Forms." *Organization Science*, Vol. 11, No. 5. (Sep. - Oct., 2000), pp. 538-550.
14. Osterloh, M., & Rota, S., (2005). "[Trust and Community in Open Source Software Production](#)," [CREMA Working Paper Series](#) 2005-11, Center for Research in Economics, Management and the Arts (CREMA).
15. Osterloh, M, & Rota, S., (2005) Open Source Software Development - Just Another Case of Collective Invention? Working Paper No. 2005 -08 Basel: CREMA
16. Prom, C., and Habing, T. (2002), "[Using the Open Archives Initiative protocols with EAD](#)." in *Proceedings of the 2nd Joint Conference on Digital Libraries*, July 14-18, 2002, edited by Gary Marchionini and William Hersch. New York: Association for Computing Machinery, pp. 171-180.
17. steve.museum, documented at <http://www.steve.museum/> (last accessed on 25 June 2007).
18. Weber, S. 2005 *The Success of Open Source*. Harvard University Press.