The Economy of Free and Open Source Software in the Preservation of Digital Artifacts

Sheila Morrissey, Senior Research Developer

ITHAKA

Sheila.Morrissey@ithaka.org

Address:

ITHAKA
100 Campus Dr., Suite 100
Princeton, NJ 08540
USA

Professional Biography:

Sheila Morrissey is Senior Research Developer at ITHAKA, a not-for-profit organization that helps the academic community use digital technologies to preserve the scholarly record and to advance research and teaching in sustainable ways. She has worked on the Portico digital preservation service, a part of ITHAKA that preserves scholarly literature published in electronic form and ensures that these materials remain accessible to future scholars, researchers, and students. She is currently engaged with partners at the California Digital Library and Stanford University in developing the next-generation JHOVE2 tool. Her past work includes the design and development of print and electronic publishing systems. She has served as a representative to XML vocabulary standards groups. She received her BA in English Literature from Yale University, her MA in English Literature from Cornell University, and her MS in Computer Science from Rutgers University.

This article original appeared in *Library Hi Tech*, Vol. 28 Iss: 2, pp.211 – 223, and is reproduced here with permission. (See http://www.emeraldinsight.com/about/policies/copyright.htm).

The Economy of Free and Open Source Software in the Preservation of Digital Artifacts

Abstract

Purpose: Free and open source software (FOSS) brings many benefits to digital preservation; however it is not "free." If we examine the context in which free and

PORTICO

open source software tools are created and employed, it becomes clear that

- The sustainability of any software (FOSS, custom or commercial) we rely on to ensure the preservation of our digital heritage will depend on careful assessment of, and provision for, the costs (implicit and explicit) entailed in the production and continued employment of these tools.

- The viability of these software tools depends on sustaining the context of these tools – in particular, in sustaining the community of knowledge and experience which makes it possible even to employ these tools.

Approach: Portico explores the costs of FOSS based on its experiences as a working archive with an extremely long time horizon.

Findings

- There are considerable benefits to FOSS, including its openness and the broad-based testing of it in real-world situations. FOSS tools can provide considerable cost savings over proprietary tools. However, FOSS is neither free to use, nor to create, nor to maintain.

- Digital preservation organizations must inventory not only the FOSS tools in our preservation arsenal, but the network of sustaining tools (FOSS and otherwise), documentation, and "tribal knowledge" that make these tools effectively usable.

- We must assess the risks to sustainability of this network of resources, and determine what it will cost to keep them viable.

- We will have to consider and implement strategies for providing the means to sustain these resources.

- An engaged community of use is the best guarantor of the vitality of any FOSS tool. As that community wanes, it becomes even more essential to capture the significant properties and domain knowledge about that tool.

- Creators of new software in the digital preservation space have a particular obligation to provide and maintain information about the significant properties of that software.

PORTICO

Value: Portico brings its practical experiences integrating multiple FOSS tools to bear on an analysis of the costs to creating and maintaining these tools over the long-term.

Keywords: FOSS, open source software, digital preservation, sustainability
Category: General Review

The Economy of Free and Open Source Software in the Preservation of Digital Artifacts

1. Introduction – The Economy, Preservation and Valuations of Software Tools

"The economy" has been much on the mind of everyone over these past eighteen months and more. For those of us in the digital preservation community, the phrase itself has become a stand-in for a chronic threat to the preservation of our digital heritage – the lack of sufficient funds. Economy – οἰκονομία – at its root the word means the management of a household. In a very old use of the term economy, the term meant the divine plan for all humanity through time (Pelikan, 1971). The original meaning of the word economist is "steward." In that sense, those of us in the digital preservation community are, fundamentally, economists.

When we speak of an economical solution to a problem, we often mean more than just one that costs the least amount of money. Software developers for example aspire to write "economical" code. In software as in mathematics, the economical solution is considered the more elegant one – the programmer's version of Occam's razor being the "KISS" principle ("Keep It Simple, Stupid"). As the discipline of economics is practiced today, the technical aspect sometimes obscures the fundamentally social and humanist activity that is, or should be, at its heart. Today's practices also obscure the many valuations – not all of them quantifiable – that comprise good stewardship (Judt, 2009).

Portico, a not-for-profit digital preservation service providing a permanent archive of electronic journals, books, and other scholarly content, makes many such valuations, as do all our colleagues in the digital preservation community. Among them is the determination of which "economical" software tools to employ in the exercise of digital stewardship. The inventory of software employed by the Portico archiving service includes a mix of custom (internally developed), commercial, and free and open source (FOSS) software. Portico, principally as a participant in the Library of Congress's National Digital Information Infrastructure & Preservation Program (NDIIPP) funded JHOVE2 project (Abrams, 2009), is a contributor to, as well as a consumer of, open source software tools. The experience of contributing to the JHOVE2 project has led us to reflect on our dual roles as open source user and con-

PORTICO

tributor, and on the economy of open source software use itself.

2. Digital Preservation and FOSS

There are many reasons why a digital preservation service might choose a FOSS tool. Principal among these is the tool's openness itself, transparency being a virtue in many aspects of digital preservation. The acquisition of most open source tools is free. And sometimes when there are no existing tools that meet the need, as in the case of JHOVE and JHOVE2, the digital preservation community must join together to create the FOSS tools it needs to accomplish its stewardship.

The complication for digital preservation, of course, is that we now have a new digital artifact. When we consider the sustainability of digital preservation, we have to be mindful that any software tool, including a FOSS tool, is itself a digital artifact, with its own preservation challenges, entailing preservations risks and requirements, raising questions of life-cycle and cost.

Consider: An archaeologist of physical artifacts typically does not need to know anything about impressing or baking clay tables before attempting to decipher a cuneiform inventory from Nineveh. Nor does a literary critic need to dust off an old clacking Underwood typewriter to study an original draft of The Maltese Falcon. But she would need software to experience, much less interpret, any digital artifact. And she would need to know how to access that software, how to use it, possibly even how to install and configure it.

The experience of a future – or indeed a current – user of FOSS is not so different from the projected beneficiary of digital preservation. If we examine the context in which these particular digital artifacts (FOSS tools) are created and employed, it becomes clear that:

- The sustainability of any software (FOSS, custom or commercial) we rely on to ensure the preservation of our digital heritage will depend on careful assessment of, and provision for, the costs (implicit and explicit) entailed in the production and continued employment of these tools.

- The viability of these software tools depends on sustaining the context of these tools – in particular, in sustaining the community of knowledge and experience which makes it possible even to employ these tools.

3. The Costs of Free

PORTICO

Let's consider the costs. What do we mean by "free" in "free and open source software?" Richard Stallman's definition is the canonical one (Free Software Foundation, 1996). These freedoms are:

- The freedom to run the program, for any purpose (freedom 0).
- The freedom to study how the program works, and change it to make it do what you wish (freedom 1). Access to the source code is a pre condition for this.
- The freedom to redistribute copies so you can help your neighbor (freedom 2).
- The freedom to distribute copies of your modified versions to others (freedom 3). By doing this you can give the whole community a chance to benefit from your changes. Access to the source code is a precondition for this.

Notably absent from his list of freedoms is the notion that FOSS is either free of cost to create, or free of cost to use.

It is commonly held that the creation of free and open software is an emergent artifact of an impassioned, self-organizing community of volunteer creators and maintainers, motivated by the intrinsic challenges of the technical problem to be solved, and philosophically committed to a communitarian ethos of software development and distribution. This is, famously, a good capsule description of the origins of Linux. It is not, of course, the case that there is no cost associated with this development model. Rather, the cost is distributed, and is reckoned in time rather than dollars. In this context, it is instructive to look at some of the estimates made by Brian Behlendorf (1999), one of the principal developers of the Apache web server, of the person-hours required to create and sustain an open-source project. He suggests that simply to maintain (i.e. not to add to or extend the functionality of) a project of moderate complexity (for example, a shopping cart plug-in) would require the equivalent of two to three full-time workers.

One might think that, the larger the installed base of a free and open source project, the less the need for monetary contributions to sustain the project. In theory an ever-widening base of volunteer contributors exists to meet the resource needs Behlendorf details. The myth of FOSS is that the only economic externality is a positive one: the network effect. Historically however, even where we have most benefited from the network effect, as with many mission-critical, widely-installed FOSS tools, it has not proved sufficient for sustaining those tools, unaided by corporate or other institutional support. It has turned out that there is also a negative

PORTICO

economic externality to FOSS: the tragedy of the commons. The "free rider" has become an unintended additional meaning of the term "free" in FOSS.

Many flagship free and open source software projects, as they have matured, have embraced some of the sustainability and revenue models similar to those recently suggested for on-line academic resources in ITHAKA S+R's recent report, Sustainability and Revenue Modes for On-line Academic Resources (Guthrie, Griffiths, and Maron, 2008). The Linux Foundation was founded essentially to provide a stable financial base of support for Linus Torvald, Linux's progenitor, and to "foster the growth of Linux." (Linux Foundation 2010c) Six corporate sponsors pay $500,000 per year; nine corporate sponsors pay $100,000 per year, and 28 corporate sponsors pay $5000 - $28,000 per year to sustain this foundation (Linux Foundation 2010a, 2010b). Corporate sponsorship of sustaining not-for-profit parent organizations has in fact become the sustainability model for most flagship FOSS projects. Thus we have the Apache Foundation, the Eclipse Foundation, the Open Source Initiative, Spring Source, OpenOffice.org, and, most recently, the WordPress Foundation.

Much FOSS software, even with corporate support, does still depend on the distributed volunteer community of contributors in various roles, as the contributor and committer lists of these projects demonstrate. We must bear in mind that we might not necessarily be able to assume a culture of generosity and communitarian spirit in a more stringent economic world – we might rather need consciously and conscientiously to take steps to foster the continuance of that culture. (Judt, 2009). As we face an era of increasing competition for increasingly scarce resources, we need to be aware of the extent to which the "free riding" there is in the free-wheeling libertarian universe of FOSS might no longer be supported by the prosperities of the larger economy. In the case of FOSS, many widely used FOSS projects are in part sustained by corporate support – support that is provided on the basis of a very cool-headed appraisal of the extent to which those projects benefit the corporate bottom line. (Casadesus-Masanell, Ramon, and Gaston Llanes, 2009; Iansiti, and Richards, 2006; Lerner and Tirole, 2002) Should any future appraisal indicate such expenditures are not in a corporation's bottom-line-focused self-interest, that support could well disappear.

Beyond the cost of creating and maintaining FOSS, there is the cost of using FOSS. There is a total cost of ownership (TCO) associated with the use of any piece of software, quite apart from the initial acquisition cost. FOSS, where it provides the required functionality, and where there is a pool of expertise in its deployment and use, provides very significant reductions in TCO, though it of course does not eliminate those costs entirely.

One of the freedoms attributed to FOSS is freedom from the costs of vendor lock-

in. The available functionality and any special formats created and used by such a tool are open to inspection and modification, and the cultural bias of the open source community toward platform independence works to prevent hardware vendor lock-in as well. However, it is not necessarily true that there would be no cost associated with switching to a different tool, whether FOSS or proprietary. It is a non-trivial operation, for example, to upgrade from one version of UNIX/Linux to another, and at least equally complicated to switch from one distribution of Linux to another (SUSE to Fedora, or DEBIAN to SUSE, for example). Indeed, an entire for-profit service industry has arisen around supporting users of FOSS tools. (See, for example, the Red Hat service and product catalog at https://www.redhat.com/wapps/store/catalog.html).

License management is another potential cost for users of FOSS. Some care must be taken, when users integrate one or more instances of FOSS into their workflows, to manage the licenses under which that software is made accessible – particularly if any customization is made to that software. The Open Source Initiative provides a list of 65 licenses they have approved via their License Review process (Open Source Initiative, 2010). Understanding these and managing compliance with their terms is a small but still real cost of use.

In considering TCO, it is also interesting to look at the British Educational Communications and Technology Agency comparison (Becta, 2005) of the TCO of open-source and proprietary software in primary and secondary schools. Schools are estimated to have saved between 24 and 44 per cent of the TCO per personal computer with FOSS compared to proprietary software across all categories (hardware, software, network, consumables, training, formal and self-support).

The Becta study is suggestive in its finding that the success of FOSS in institutional settings varies significantly as a result of what might be termed cultural rather than technical causes. Schools with a strong and experienced advocate for FOSS showed best results for satisfaction of institutional needs. Indeed, there is a great deal of evidence that context, the community of knowledge and experience, is as critical as financial resources for the creation and use of these tools. It is instructive to look at the "experience at the front" detailed in articles of an earlier volume of this journal to view at ground level some of the complications of analyzing, selecting, and integrating FOSS into an institution's process and IT infrastructure (Donnelly, 2010; Garza, 2009; Huttenlock, Beaird, and Fordham, 2006). These experiences bear out the observations of Garlan, Allen and Ockerbloom (2009) that "reuse is still so hard." While there are some excellent protocols for evaluating and selecting FOSS (see, for example, OSS Watch (2008), it is notable that in actual practice it is often the case that "first fit" rather than "best fit" is the determining criterion – most especially if there is a reasonable fit with a FOSS application with which someone in the selecting institution has some experience (Hauge, Osterlie, Sorensen, and

PORTICO

Gerea, 2009).

4. FOSS Case Study: JHOVE2

Portico's experience developing JHOVE2 certainly bears out this notion of the critical nature of context and community in FOSS development and use. JHOVE2 is comprised of a modular framework into which functional components can be plugged to provide digital object characterization: identification, validation, feature extraction, and assessment (Abrams, 2009).

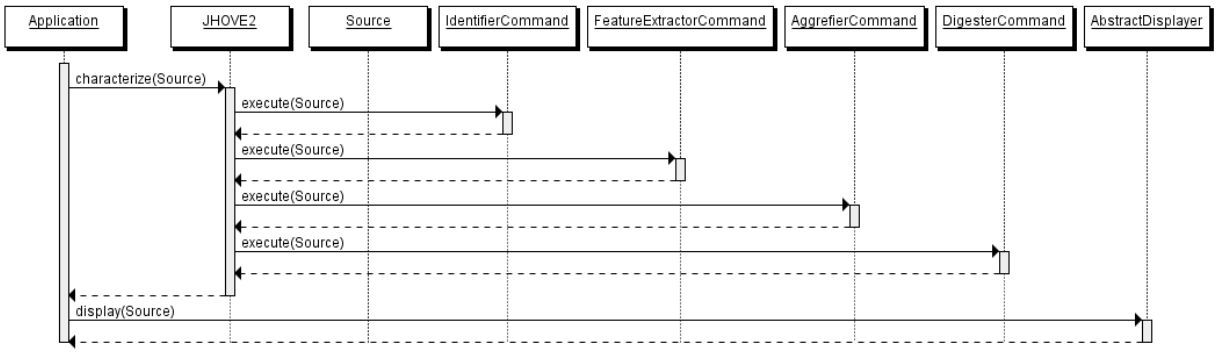

Figure 1
JHOVE2 (Beta) Characterization Sequence Diagram
(Abrams, Anderson, Frost and Morrissey, 2009)

One of JHOVE2's architectural design principles is ease of integration of existing code bases to extend its functionality – in particular, to extend the range of format families upon which it can perform characterization. Thus the JHOVE2 identification module comprises a wrapper around the UK National Archives DROID identification tool (http://sourceforge.net/projects/droid/). The framework and identification modules have been constructed in such a way that other developers in the JHOVE2 community could create a similar wrapper around a different identification tool (whether an existing tool such as BSD File, or a tool created from scratch), and plug that new identifier into the JHOVE2 framework in place of DROID.

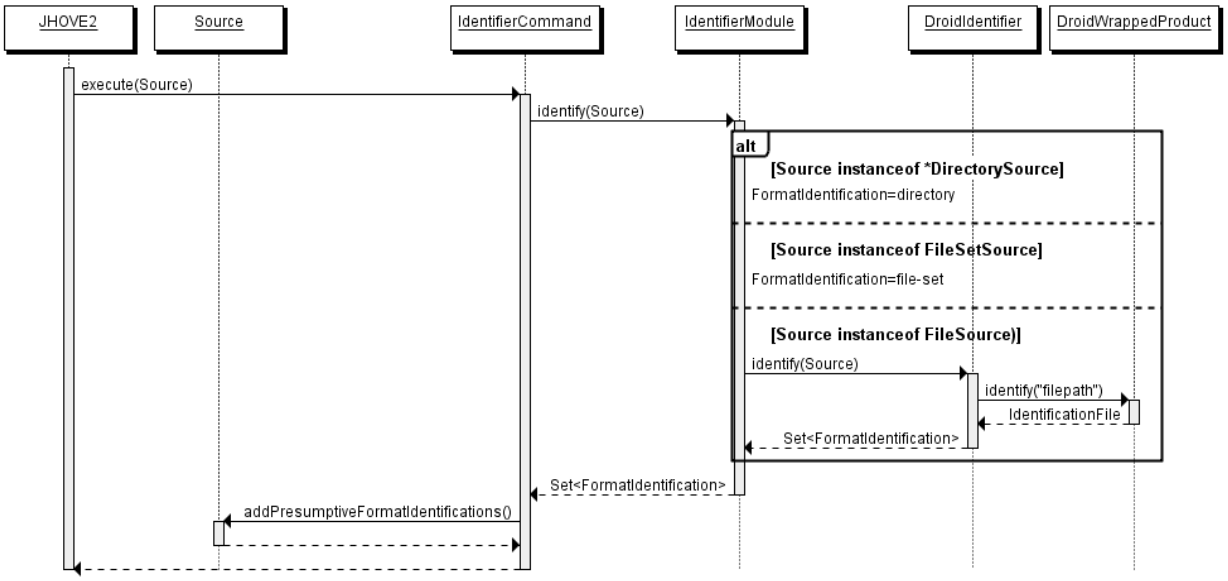

Figure 2
JHOVE2 (Beta) Identification Sequence Diagram
(Abrams, Anderson, Frost and Morrissey, 2009)

To perform validation and feature extraction on SGML files, the JHOVE2 SGML module will integrate the functionality of James Clark's Open SP tool (Clark). SGML has its origins at IBM in 1969 (Goldfarb, 1996), and became an ISO standard (ISO, 1986) in 1986. While not obsolete, it is certainly at least venerable. With the exception of HTML, most SGML applications have given way over the period since the publication of the XML standard in 1998 to XML, for which it has proven considerably easier to provide a toolkit of parsers, transformation languages, and query languages. There are very few commercial SGML tools available at present. James Clark's Open SP tool, noted for its exceptional coverage of the SGML feature set, is fortunately available as FOSS.

Open SP is written in C++, which had its origins in 1979 (Stroustrup, 1995). While by no means an obsolete language, it is no longer the lingua franca of the computer science world. The AP exam in computer science, for example, which tested students knowledge of C++ from 1999-2003, has switched to testing their knowledge of JAVA since 2004 (http://en.wikipedia.org/wiki/Advanced_Placement_Computer_Science.) JHOVE2 is written in JAVA, and must somehow communicate with this C++ tool.

From a digital preservation perspective, the implementation of a JHOVE2 SGML tool provides us an interesting experience both of tools and of domain knowledge that

PORTICO

are beginning to recede from present use. What does it take to make active use of an older FOSS tool on an instance of an increasingly obsolescent format within another FOSS tool?

The executable Open SP code for Windows that is available today for download is built for the Windows 95 and Windows NT operating systems – both considerably out of date. A user of JHOVE2 on more recent Windows platforms will therefore need to download this application as source code. The instructions for creating executable code from the source code expect a user to have a Microsoft Visual C++ Version 6.0 compiler. This proprietary tool is four generations old – dating from 1998.

A current (2010) user without the proprietary and now-obsolete Microsoft compiler can, nevertheless, create a running version of this tool, using still more free and open source software. In capsule form, at Portico we accomplish this by:

1. Downloading and installing the Eclipse Integrated Development Environment

2. Downloading and installing the Eclipse C++ Development Toolkit (CDT)

3. Downloading and installing MinGW (A native Windows port of the GNU Compiler Collection (GCC), with freely distributable import libraries and header files for building native Windows applications), taking care to install a version that is compatible with the version of the Eclipse CDT previously installed

4. Installing the g++ compiler

5. Installing the gdb debugger

6. Installing the MSYS make utility

7. Downloading Open SP

8. Importing Open SP into Eclipse and building the application

What is first notable about this protocol is the rich toolset of FOSS available for use. This is both a great benefit (we do not have to resort to proprietary software) and an additional preservation conundrum: in order successfully to use a single FOSS tool, such as Open SP, it is necessary to maintain an ecosystem of FOSS tools Additionally, this capsule list is a digest of many sources of information (all freely and openly available on the Internet), of varying quality and reliability, referring to various versions of the pieces of software used to assemble the tool, discussing problems encountered configuring the tools as described, sometimes, and sometimes not, giving solutions when the instructions turn out not to work. Somehow this

PORTICO

context, this repository of experience in the configuring and use of these tools, also must be preserved.

We now have an executable available for use. However, integration with JHOVE2 is complicated by the fact that the JHOVE2 framework and the Open SP tool were written in different languages. This means that integrating the executable is somewhat more complicated than would be the use of another JAVA tool. Either the integration code must make use of a tool called the Java Native Interface (JNI) to invoke the Open SP tool, or it must "shell out" and run the tool as a stand-alone piece of code, and then process the output the native C++ executable produces (This latter method was used by Portico when creating an SGML module for the earlier JHOVE characterization tool). So an additional requirement of the archaeological use of FOSS is the use of well-defined and well-understood interfaces in the FOSS tool, and access to a library for adapting those interfaces to use in another software context. This so-called "adapter pattern" is a well-understood software engineering approach. The implementation however, across different software language boundaries, can be fraught with difficulties, as experience with JNI has borne out. In the approach taken by Portico, successfully using the tool means a deep understanding of the semantics of the intermediate output created by Open SP, which in turn depends on access to the documentation of the output format. This documentation, again, is freely available on the Internet, and again constitutes an additional item to be preserved to guarantee the integrity of the web of information required to make use of this digital artifact.

One of the benefits of the transparency of open source software is that "given enough eyeballs, all bugs are shallow." (Raymond, 2000) Because Open SP is open-source, Portico was able to repair a minor bug that required fixing. As the code is no longer under very active development, it was important that Portico was able to do so. But doing so again required an ecosystem not just of tools but of knowledge, experience, documentation – in coding languages (C++), in the development environments and the "build" tools described above, in the coding idioms of the original developer, and in what is called domain knowledge – knowledge of SGML, of the SGML standard, of commentaries on some of the opacities of that standard – all of which are needed simply to understand the semantics of the command-line options of Open SP. While much of this information, including the SGML specification, is still available on-line, many of the links to information about SGML are already broken. Portico has the advantage of a pool of developers and others with direct experience with SGML applications (and with personal libraries of now out-of-print books about SGML). But this is a community of resources that one would reasonably expect to diminish over time.

Young as it is as a discipline, software engineering has a history, and conventions and practices that have evolved over time. A condensed history of those conven-

tions and practices would begin with people flipping binary switches, then coding in assembler languages, then using Fortran, COBOL, and other procedural languages, then refining those procedural languages with techniques of structured programming. The taxonomic tree would branch out into object oriented idioms and languages, functional programming, declarative programming, resolution theorem proving logic programming, the use of non-deterministic and genetic algorithms in machine learning, and, cutting across many of these, the use of massively parallel programming idioms. It is not only particular programming languages that have a vogue and then disappear from common use; so do the large-scale conceptual structures of software construction. And the community of users schooled in these various languages and idioms fluctuates – and can diminish to the vanishing point – over time.

Portico's experience with Open SP suggests that the "openness" of FOSS is not just a feature – it is an active demand on its users. We will continue to need the skills not just to configure and use, but to read our way through these FOSS tools. And we will need more contextual information than the tools themselves provide in their code, even to discern what might be problematic with their use. Perhaps the best exemplar of this is the Y2K bug. In its original context, of course, the Y2K bug was the Y2K design feature: it was a technique to conserve what was, at the time, very expensive memory and disk storage space. It was, in fact, a short-term economy with long-term economic consequences. But, more importantly, in the context in which the software was intended to be used, and under the governing assumptions in which it was written, it was not a bug at all. We must be prepared for problems like this to recur. Users of software that relies on so-called "Unix time" will have to compensate for the so-called "Year 2038 problem," (Krill, 2010), 19 January 2038 being the latest date that can be represented in older UNIX architectures and the C and C++ programs written to those architectures. Users of the date and time C/C++ libraries have not necessarily coded defensively around this upper-bound limitation of these libraries, even if they were aware of it. Effective knowledge of the constraints in the use of such software means maintaining not just the static documentation of the C/C++ programming libraries, but also a living awareness in the community of use of these artifacts about potential issues with temporal data structures in older architectures and coding implementations. Any individual FOSS tool might not explicitly detail these constraints.

5. Implications and Conclusions

The general implication, of course, is that we cannot assume sufficient information will be encapsulated in the source code of a particular FOSS tool to sustain the effective usefulness of that tool. A richer network of knowledge and experience is necessary as well. As Matthews, Bicarregue, Shaon and Jones (2009) have pointed out, the issue of encapsulating sufficient knowledge for use and reuse of software is

not solely a digital preservation concern. It is adumbrated within the best practices of software engineering as a discipline. Initiatives such as OSGi, for example, or the built-in dependency-chain resolution capability of the Java MAVEN tool, are pragmatic responses to a small subset of this problem, the resolving of all the correctly-versioned Java components required to assemble a Java application with dependencies on those components.

The JISC study on the significant properties of software (Mathews, et. al., 2008) does an excellent job of developing taxonomy of what might constitute a minimal information set of significant properties for re-instantiation of a software instance, including an articulation of the network of software and other dependencies required. The study authors have also provided a prototype tool (Shaon and Bicarregui, 2009) for associating descriptors of these properties with a Java project within the Eclipse development environment. Software developers as a group are famously less than enthusiastic about providing even this minimal subset of documentation for the software they write. This means that moving this tool beyond proof of concept would mean automating information collection as much as possible. Even such a taxonomy and such a tool, however, would have a fair chance of missing such "significant properties" as are exampled by the Y2K problem. For the near term, at least, the "human in the loop" will be essential for the effective use of FOSS tools. There will always be a certain amount of archaeology – reverse engineering, or more simply, reading of code – to be done.

What does this experience of Portico and others in the creation and use of FOSS suggest to us as stewards of these particular digital assets? What is entailed in their economic use?

- We need to bear in mind that the considerable benefits of FOSS – its openness, the broad-based testing of it in real-world situations – while typically providing considerable cost savings over proprietary tools, is neither free to use, nor to create, nor to maintain.

- We need consciously to inventory not only the FOSS tools in our preservation arsenal, but the network of sustaining tools (FOSS and otherwise), documentation, and "tribal knowledge" that make these tools effectively usable.

- We have to assess the risks to sustainability to this network of re sources, and determine what it will cost to keep them viable.

- We will have to consider and implement strategies for providing the means to sustain these resources.

- An engaged community of use is the best guarantor of the vitality of any FOSS tool. As that community wanes, it becomes even more essential to capture the significant properties and domain knowledge about that tool.

PORTICO

- Creators of new software in the digital preservation space have a particular obligation to provide and maintain information about the significant properties of that software.

The undertakings of several free and open source digital preservation software initiatives to establish that software on a sustainable economic basis (for example, the work of the MetaArchive or DuraSpace), are leading indicators of the maturing recognition that FOSS itself comes under the umbrella of digital preservation, and that strategies to ensure its sustainability are critical to the whole enterprise of "interoperability with the future." We will all benefit from the experience of the developers of Dioscuri emulator project (van der Hoeven, van Wijngaarden, Verdegem and Slats, 2005) in developing intuitions about how wide or shallow a net we must cast around software tools, associated documentation, and undocumented domain knowledge in order to use FOSS tools effectively going forward. As we gain experience in determining the limits to the amount of information that can reasonably be encapsulated in an individual FOSS tool, we may discover a new role for digital libraries and librarians, as loci of experience and effective knowledge in the assembly and application of these FOSS tools.

Finally, we need to be sensitive to the potentially inevitable decline and possible disappearance of context, of a community of knowledge and use, necessary for the effective use of certain FOSS tools, or the effective reconstitution of some digital artifacts. Chris Rusbridge's (2006) key insight, that digital preservation is "a series of holding positions, or perhaps a relay," comes very much to mind here. At first blush this might seem to make digital stewardship a sort of housework, done today, and needing to be done again tomorrow. But perhaps "housework" is no bad description for our "economy:" the truly economical management of our digital household.

Acknowledgements

The JHOVE2 project is funded by the Library of Congress as part of its National Digital Information Infrastructure Preservation Program (NDIIPP). The author would like to acknowledge the contributions to this paper of Eileen Fenton, Executive Vice President, Content and Technology Services and Portico Managing Director, ITHAKA; of Amy Kirchoff, Archive Service Product Manager, ITHAKA; and of John Meyer, Director of Data Technology, ITHAKA.

References

Abrams, Stephen, Morrissey, Sheila and Cramer, Tom (2009), "What? So What?: The Next-Generation JHOVE2 Architecture for Format-Aware Characterization", The

PORTICO

International Journal of Digital Curation, Vol. 4, No. 3, available at: www.ijdc.net/index.php/ijdc/article/view/139 (accessed 13 January 2010)

Abrams, Stephen, Anderson, Richard, Frost, Hannah and Morrissey, Sheila (2009), "'What? So What?': JHOVE2 Next-Generation Characterization Public Workshop in Conjunction with the iPRES 2009 Conference", available at: https://confluence.ucop.edu/display/JHOVE2Info/Project+Presentations (accessed 13 January 2010)

Anderson, Paul (2006), "Crossing the Chasm: open source software comes of age", available at: www.oss-watch.ac.uk/resources/sustainability06.xml (accessed 13 January 2010)

Anderson, Paul (2008), "Leveling the playing field: developing a mixed economy for software procurement", available at: www.oss-watch.ac.uk/resources/procurement08.xml (accessed 13 January 2010)

Behlendorf, Brian (1999), "Open Source as a Business Strategy", in DiBona, Chris and Ockman, Sam (Ed.), Voices from the Open Source Revolution, O'Reilly Media, Chapter 11, available at: http://oreilly.com/catalog/opensources/book/brian.html (accessed 13 January 2010)

Boyd, Robert (2008), "Staffing the Commons", Library Hi Tech, Vol. 26, No. 2, pp. 232-243

British Educational Communications and Technology Agency (Becta) (2005), "Research report: Open source software in schools: A study of the spectrum of use and related ICT infrastructure costs", available at: http://publications.becta.org.uk/display.cfm?resID=25907 (accessed 13 January 2010)

Casadesus-Masanell, Ramon, and Gaston Llanes. (2009), "Mixed Source", Harvard Business School Working Paper, No. 10-022, September 2009, available at: www.hbs.edu/research/pdf/10-022.pdf (accessed 13 January 2010)

Clark, James, "SP: An SGML System Conforming to International Standard ISO 8879 -- Standard Generalized Markup Language", available at: www.jclark.com/sp/ (accessed 13 January 2010)

Donnelly, Francis Patrick (2010), "Evaluating Open Source GIS for Libraries", Library Hi Tech, Vol. 28, Issue 1, preprint available at: www.emeraldinsight.com/Insight/viewContentItem.do;jsessionid=E53DB9E089BA51F3C88A2CC38C346A82?contentType=Article&contentId=1827513

Dougiamas, Martin (2007), "Moodle: a case study in sustainability", available at:

PORTICO

www.oss-watch.ac.uk/resources/cs-moodle.xml (accessed 13 January 2010)

Free Software Foundation (1996), "The Free Software Definition", available at: www.gnu.org/philosophy/free-sw.html (accessed 13 January 2010)

Gardler, Ross (2008), "Sustainable open source", available at: www.oss-watch.ac.uk/resources/sustainableopensource.xml (accessed 13 January 2010)

Garlan, David, Allen, Robert and Ockerbloom, Robert (2009), "Architectural Mismatch: Why Reuse is Still So Hard", IEEE Software, Vol. 26, No. 4 , July/August 2009, pp. 66-69

Garza, Alejandro (2009), "From OPAC to CMS: Drupal as an extensible library platform", Library Hi Tech, Vol. 27, Issue 2, pp.252 – 267

Goldfarb, Charles F. (1996), "The Roots of SGML", available at: www.sgmlsource.com/history/roots.htm (accessed 13 January 2010)

Guthrie, Kevin, Griffiths, Rebecca and Maron, Nancy (2008), "Sustainability and Revenue Models for Online Academic Resources: An Ithaka Report", available at: www.ithaka.org/ithaka-s-r/strategy/sca_ithaka_sustainability_report-final.pdf (accessed 13 January 2010)

Hall, James (2009a), "Open Source and Free Software", available at: http://scienceblogs.com/collectiveimagination/2009/10/james_hall_on_free_and_open_so.php (accessed 13 January 2010)

Hall, James (2009b), "Open Source Software in the Real World", available at: http://scienceblogs.com/collectiveimagination/2009/10/james_hall_open_source_softwar.php

Hall, James (2009c), "Cultivating Open Source Software", available at: http://scienceblogs.com/collectiveimagination/2009/10/cultivating_open_source_softwa.php (accessed 13 January 2010)

Hauge, O., Osterlie, T., Sorensen, C., and Gerea, M. (2009), "An empirical study on selection of Open Source Software - Preliminary results", appears in Proceedings of the 2009 ICSE Workshop on Emerging Trends in Free/Libre/Open Source Software Research and Development (May 18, 2009), International Conference on Software Engineering, IEEE Computer Society, Washington, DC, pp, 42-47

Huttenlock, Terry L., Beaird, Jeff W. and Fordham, Ronald W. (2006), "Untangling a tangled web: a case study in choosing and implementing a CMS", Library Hi Tech,

PORTICO

Vol. 24, No. 1, pp. 61-68

Iansiti, Marco, Ph.D. and Richards, Gregory L., (2006), "The Business of Free Software: Enterprise Incentives, Investment, and Motivation in the Open Source Community", appearing in Harvard Business School Working Paper Series, No. 07-028, 2006, available at: www.hbs.edu/research/pdf/07-028.pdf (accessed 13 January 2010)

ISO (International Organization for Standard) (1986), "ISO 8879:1986 Standard Generalized Markup Language (SGML)", available at: www.iso.org/iso/catalogue_detail.htm?csnumber=16387 (accessed 13 January 2010)

Judt, Tony (2009), "What is Living and What is Dead in Social Democracy", The New York Review of Books, Vol. 56, No. 20, available at: www.nybooks.com/articles/23519 (accessed 13 January 2010)

Katsamakas, E. (2007), "Why Most Open Source Development Projects Do Not Succeed", appears in: First International Workshop on Emerging Trends in FLOSS Research and Development, 2007. FLOSS '07, pg. 3

Krill, Paul (2010), "Y2K: 10 Years Later", Infoworld, January 04, 2010; available at: www.infoworld.com/d/adventures-in-it/y2k-10-years-later-459?source=footer (accessed 13 January 2010)

Lerner, Josh and Tirole, Jean (2002), "Some Simple Economics of Open Source", Journal of Industrial Economics, Vol. 50, Issue 2 (June 2002), pp. 197-234.

Linux Foundation (2010a), "Corporate Membership", available at: www.linuxfoundation.org/about/join/corporate (accessed 13 January 2010)

Linux Foundation (2010b), "Members", available at: www.linuxfoundation.org/about/members (accessed 13 January 2010)

Linux Foundation (2010c), "About Us", available at: ww.linuxfoundation.org/about (accessed 13 January 2010)

Matthews, Brian, Brian McIlwrath, David Giaretta and Esther Conway (2008), "The Significant Properties of Software: A Study", available at: http://www.jisc.ac.uk/media/documents/programmes/preservation/significantpropertiesofsoftware-final.doc

Matthews, Brian, Bicarregue, Juan, Shaon, Arif and Jones, Catherine (2009), "Framework for Software Preservation", available at: http://epubs.stfc.ac.uk/bit-

PORTICO

stream/4059/SoftwarePreservationFramework-final.pdf

Metcalf, Randy (2004), "Top Tips For Selecting Open Source Software", available at: www.oss-watch.ac.uk/resources/tips.xml (accessed 13 January 2010)

Mutula, Stephen Mudogo and Kalaote, Tumelo (2010), "Open Source Software Deployment in the Public Sector: A Review of Botswana and South Africa", Library Hi Tech, Vol. 28, Issue 1 (preprint)

Open Source Initiative (2010), "Licenses by Name", available at: www.opensource.org/licenses/alphabetical (accessed 13 January 2010)

OSS Watch (2008), "Decision factors for open source software procurement", available at: www.oss-watch.ac.uk/resources/procurement-infopack.xml (accessed 13 January 2010)

O'Mahoney, Siobhan (2005) "Nonprofit Foundations and Their Role in Community-Firm Software Collaboration" in Feller, Joseph, Fitzgerald, Brian, Hissam, Scott A. and Karim R. Lakhani (Ed.), Perspsectives on Free and Open Software, MIT Press, Cambridge, MA, pp.425-447

Pelikan, Jarislav, (1971) The Christian Tradition Volume 1 The Emergence of the Catholic Tradition (100-600), University of Chicago Press, Chicago and London, pp.228-229

Raymond, Eric S. (2000), "The Cathedral and the Bazaar", available at: www.catb.org/esr/writings/cathedral-bazaar/cathedral-bazaar/ (accessed 13 January 2010)

Rothenberg, Jeff (1998), "Avoiding Technological Quicksand: Finding a Viable Technical Foundation for Digital Preservation", available at: www.clir.org/pubs/reports/rothenberg/contents.html (accessed 13 January 2010)

Rothenberg, Jeff (2000), "An Experiment in Using Emulation to Preserve Digital Publications", available at: http://nedlib.kb.nl/results/emulationpreservationreport.pdf (accessed 13 January 2010)

Rusbridge, Chris (2006), "Excuse me: Some Digital Preservation Fallacies?", Ariadne, Issue 46, available at: www.ariadne.ac.uk/issue46/rusbridge/ (accessed 13 January 2010)

Seadle, Michael (2008), "The digital library in 100 years: damage control", Library Hi Tech, Vol. 26, No. 2, pp. 5-10

PORTICO

Shaon, A. and Bicarregui, J.C. (2009), "Tools and Guidelines for Preserving and Accessing Software as a Research Output Report III: Tools", available at: http://epubs.stfc.ac.uk/bitstream/4135/SPEQS_Report.pdf

Shaon, A., Woodcock, J. and Conway, E. (2009), "Tools and Guidelines for Preserving and Accessing Software as a Research Output Report II: Case Studies", available at: http: //epubs.stfc.ac.uk/bitstream/4134/casestudies-report.pdf

Stark, Mallory (2003), "The Organizational Model for Open Source: Q&A with Siobhan O'Mahoney", available at http://hbswk.hbs.edu/item/3582.html (accessed 13 January 2010)

Stroustrup, Bjarne (1995), "A History of C++: 1979-1991", available at: www2.research.att.com/~bs/hopl2.pdf (accessed 13 January 2010)

van der Hoeven, Jeffrey, van Wijngaarden, Hilde, Verdegem, Remco and Slats, Jacqueline (2005), "Emulation: a viable preservation strategy", available at: www.kb.nl/hrd/dd/dd_projecten/Emulation_research_KB_NA_2005.pdf (accessed 13 January 2010)

Wilson, James A. J. (2006), "Open Source Maturity Model", available at: www.oss-watch.ac.uk/resources/osmm.xml (accessed 13 January 2010)

Wilson, Rowan (2005), "Open source development - an introduction to ownership and licensing issues", available at: www.oss-watch.ac.uk/resources/iprguide.xml (accessed 13 January 2010)

Zawinski, Jamie (1999), "resignation and postmortem", available at: www.jwz.org/gruntle/nomo.html (accessed 13 January 2010)