eResearch workflows for studying free and open source software development

James Howison, Andrea Wiggins and Kevin Crowston

School of Information Studies
Syracuse University
{jhowison|awiggins|crowston}@syr.edu

Abstract. This paper proposes a demonstration of eResearch workflow tools as a model for the research community studying free and open source software and its development. For purposes of background and justification, the paper first introduces eResearch as increasingly practiced in fields such as astrophysics and biology, then contrasts the practice of research on free and open source software. After outlining the suitable public data sources the paper introduces a class of tools known as scientific workflow frameworks, specifically focusing on one—Taverna—and introducing its features. To further explain the tool a complete workflow used for original research on FLOSS is described and the agenda for the live demonstration is outlined.

1 Background: eResearch

eResearch[1] refers to a set of scientific practices and technologies which allow distributed groups of scientists to bring to bear large shared data sets, computational resources and shared workflows for scientific inquiry. A prototypical example of such research is the Upper Atmospheric Research Collaboratory (UARC) [1] and the NSF has produced a series of reports on the topic [2–4].

The hallmarks of eResearch are:

- broad community-level collaborations between distributed scientists
- large-scale broadly available data sets
- shared computational analysis tools and workflows
- replicable research with clear provenance metadata

While the FLOSS research community has taken some steps towards this goal we have not yet fully embraced this model of inquiry, as highlighted below. Given that we study highly effective distributed collaborators and collaborative technologies in the FLOSS community, we have good understanding of the

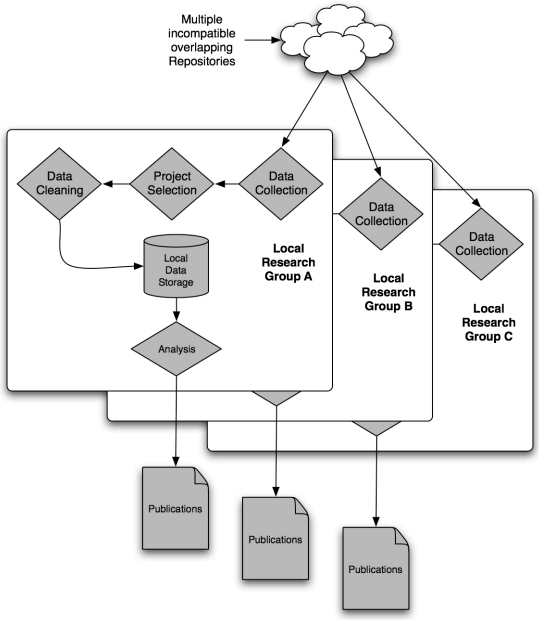
[1] eResearch is also often referred to as Cyberinfrastructure (CI) or eScience and eSocialScience as well as Scientific Collaboratories. The techniques and organization described in this paper are applicable across fields of inquiry from the natural sciences, the social sciences and engineering, therefore this paper adopts the most general term: eResearch.

challenges of, and solutions to such distributed collaborations. It is time to build further towards it.

Of course, the Software Engineering community has not been entirely absent from these changes in scientific practice. The PROMISE archive in Software Engineering collects data sets designed to help create predictor models, such as understanding which modules are most likely to develop defects [5, 6]. The PROMISE data includes a well-respected set from NASA and a number of open source software projects. They do not collect data themselves, but re-publish the data collected by research teams, together with metadata about the data. The field has also built many tools, such as those which measure software complexity, but is yet to make substantive use of workflow technologies which bind data and analyses together for replicable and extendable scientific inquiry.

1.1 Current FLOSS research practices

Fig. 1. FLOSS researchers often unnecessarily repeat data collection



A series of papers has examined the current research practices in the investigation of FLOSS and its development [7–10]. In general this research has been, and continues to be, undertaken by separate groups and has involved substantial re-collection of very similar data, usually through spidering Sourceforge or similar sites. Figure 1, from [8], shows the situation as it was in 2005.

Since then significant progress has been made towards shared datasets held in what have been called Repositories of Repositories (RoRs) [10]. At the IFIP 2.13 conference in 2007 a workshop was held for research based on public

databases, including FLOSSMole and CVSanalY. In addition there are the Notre Dame Sourceforge database dumps, available under an academic sublicense. These data sources provide an excellent foundation for moving research in this field towards eResearch. Figure 2 summarizes the impressive and substantial amount of data available in these RoRs.

Of course not all researchers use these databases, with many continuing to spider their own data sets, especially outside communities such as IFIP 2.13 and the Mining Software Repositories workshops. For example at the International Conference on Information Systems in 2007 there were over five papers using independently spidered FLOSS data.

Figure 2 also indicates substantial gaps, some of which may be filled if currently planned work (colored grey) is completed. For example, currently there is no easy comprehensive source of mailing lists for projects, nor are there current plans to archive project's IRC communications or to understand the structure of dependencies between projects (perhaps by parsing the Debian package dependancies or examining library usage or code reuse, as studied in [11]).

Perhaps more significantly there is a myriad of project specific information discovered by researchers in the course of their research which is not contained in these databases. For example we do not yet have a way to collect more qualitative information which may be crucial for interpreting these large data sources. For example, while FLOSSmetrics is making a great start by manually validating repository locations, it is clear that projects move between hosting technologies so it would be useful to know where projects hosted their main distribution channel or source code repository at different points in its lifetime. Or whether the project uses a patch submission procedure for its peripheral developers or simply grants SCM access for all patches? Or which sets of projects can be analyzed as direct competitors (thus allowing more theoretically meaningfully use of measures such as downloads).

While progress has been made in data availability there is little sharing of analyses, including components that could provide or calculate important measures such as project effectiveness. In general researchers have stuck to their (or perhaps to their graduate students) preferred data manipulation and statistical analysis tools, conducting in-house development where required. Those researchers which have made their analysis components and workflows available (such as [12] and [13]) have merely placed such bespoke tools on project websites.

This paper proposes the demonstration of tools that can move research on FLOSS towards increased collaboration and better research results. Figure 3 shows an envisioned workflow repository which allows the discovery, replication, extension and publication of research workflows, drawing on shared components.

Fig. 2. Table showing publicly available FLOSS data for research

	Repository for Research Data[1]:	FLOSS mole	Notre Dame dumps	FLOSS metrics & CVSanalY	Qualoss & SQO-OSS	Source kibitzer
Project Demographics[2]	Basic data	■	■	■		
	Confirmed Locations			■		■
Developer Demographics	Memberships	■	■			■
	Roles	■	■			■
Communication Venues	Mailing lists	▒		▒		
	Forums	▒	■			
	Issue Trackers	▒	■	▒		
	IRC logs					
	Release System	■	■			
Software Venues	SVN/CVS (counts)	■	■	■		
	SVN/CVS full			■		
	Packages produced	■	■	■		
	Releases + Dates	■	■	■		
	Size (LOC, SLOC)			■	▒	■
	Dependencies					
	Complexity Metrics				▒	■
Use and Popularity	Downloads	■	■			
	Pageviews	■	■			
	User ratings	■				
	In Debian	■		■		
	Actual Use[3]	▨				
Sample Collected	Sourceforge	■	■	▨	See Note 5	See Note 4
	Rubyforge	■				
	ObjectWeb	■		▨		
	Savannah (GNU)			See Note 6		
	Debian Distribution	■				
	Apache Foundation			▨		
	GNOME meta-project			▨		
	KDE meta-project				▒	

□ Not Collected ▒ Planned (or pilot data only)
▨ Partial (selected sub-collection) ■ Present

1. This table excludes services with data not easily available to researchers. Ohloh, for example, was excluded for this reason. The Notre Dame dumps require signing a research usage agreement. Sourcekitbizer was included insofar as it provides public access to data via the FLOSSmole project. FLOSSmetrics includes the earlier sets released by the Libre Software engineering group (CVSanalY and Debian Counts). Qualoss and SQO-OSS are included together for reasons of space, they are separate projects, but they are collaborating.

2. Project Demographics include Names, Descriptions, Founding date, Intended Audience, Operating System/environment, License, Programming language, Maturity/Status and Donors. Projects are often hosted on more than one service, or provide their own services (such as Trac, SVN etc) Confirmed Locations refers to a human effort to identify the locations actually used by each project.

3. Actual use as measured, for example, by the Debian Popularity contest which has a voluntary agent installed by some Debian users that reports frequency of package use.

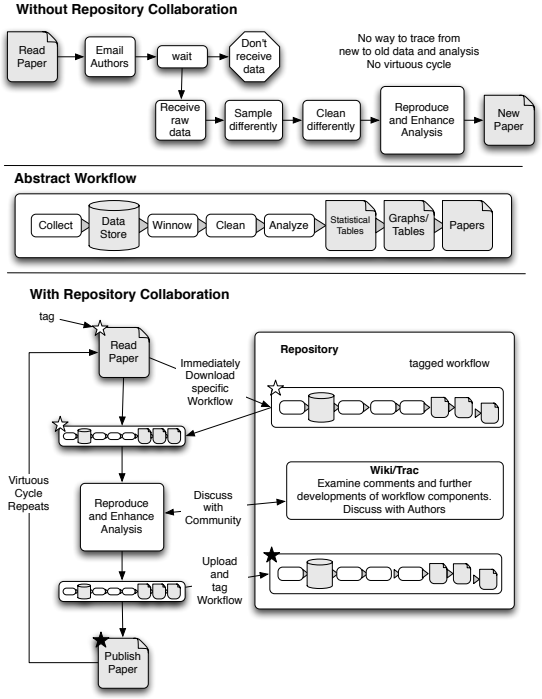
4. Sourcekibitzer samples only Java projects and accepts user contributions (specify project, SCM location, homepage)

5. Qualoss intends to implement their measures on 50 projects, currently there are 5 available as pilot data. SQO-OSS works closely with the KDE meta-project.

6. FLOSSmetrics aims to have validated data for 3,000 and currently has partial data available (primarily CVSanalY) for 100 projects.

URLS: FLOSSmole: ossmole.sf.net, Notre Dame: nd.edu/~oss/, FLOSSmetrics: data.flossmetrics.com, CVSanalY: libresoft.es/Results/CVSAnalY_SF, Qualoss: qualoss.org, SQO-OSS: www.sqo-oss.eu, Sourcekibitzer: sourcekibitzer.org. Thanks to Jesus González-Barahona, Gregorio Robles and Megan Conklin for assistance in preparing this table.

Fig. 3. Envisioned improvements in FLOSS research practices



2 Scientific workflow tools

There are a set of tools which contribute to solving the issues raised above. This section introduces two: workflow tools and a community platform for distributed collaboration around them.

2.1 Workflow tools

Workflow tools support high-level programming which binds together data sources and analysis steps. Known collectively as "scientific workflow tools" examples include Taverna[2] and Kepler[3]. The basic principles of the software are the same: steps in a workflow are undertaken by components which have multiple input and output ports. Components are linked by joining the output ports of one to the input ports of another. A workflow made up in such a manner can be represented simply as a flow diagram (See Figure 5, below) and is usually realized in a single xml file. As with most programming environments,

[2] `http://taverna.sourceforge.net`
[3] `http://kepler-project.org`

much of the usefulness of these tools comes from their library of components which can be local (eg Java or *R*) or remote (eg SOAP accessed web-services).

The workflow tool to be demonstrated allows the creation of components in any language, by supporting interfaces such as SOAP or command-line execution. However the tool does make the use of some technologies easier and more portable. The high-level composition also promotes modularity in software development, believed to lead to easier collaboration and higher quality software. Though not perfect, see discussion below, iodeally workflow system leaves individual researchers free to innovate while providing a platform to encapsulate and share their innovations.

Different scientific communities have begun to coalesce around particular tools, with Taverna being particular used in the life-sciences, such as biology and chemistry) and Kepler in sciences such as astro-physics, geology and ecology. Recently lightweight platforms have emerged to share workflows (and their embedded components), drawing on social-networking models like MySpace and Facebook (these are discussed further below).

Taverna The proposed demonstration will focus on Taverna and will use workflows developed to address research questions about FLOSS development. Taverna is instantiated as a stand-alone desktop application, written in Java and therefore able to run on a wide selection of operating systems including Windows, Mac OS X, Linux and other Unixes operating systems. It is also possible to run workflows created with Taverna via a 'headless' application for server-based unattended runs.

The application has two main interface modes: one for the design of a workflow and one for its execution. The design mode consists of three sections: available components, the workflow with the selected components and named ports, and an automatically drawn diagram showing the workflow. Unfortunately connections are made through right-clicking the selected components and not by drag-and-drop in the graphical depiction, but the process is relatively simple once learned. Input and output ports are typed through the familiar MIME typing system. For simplicity and sharing, components can be grouped into sub-workflows, with a single set of input and output ports.

There are two main classes of components: remote and local. Remote services can be gathered ('scavenged' in Taverna parlance) by entering URLs including Web Services Description Language (WSDL), for SOAP accessed services, and a format developed in the biology community called 'biomoby'. The demonstration will utilize remote services created by the authors which present a WSDL/SOAP interface, run from a Ruby on Rails backend server. It is anticipated that the development of remote services in the FLOSS domain will utilize the WSDL/SOAP combination, since it is standard in many server technologies (through libraries such as Axis for Java). The application parses the WSDL description file and makes available multiple components, each with their named input and output ports.

Local services include a set of already written components dealing with standard operations such as file IO, string and list manipulation. It is also possible to write customized local components in Java and the statistical programming language R[4]. Java components are actually written using Beanshell, which is a scripting language simplification of Java. Data fed into a components Input ports are available as local variables of the same name and, similarly, output is automatically taken from variables with the same name as the output port at the end of a scripts execution. Both R and Beanshell scripts are stored in the workflow file, and one can parse a workflow file to make these components available for re-use. R components require an instance of Rserve running either locally or on a server. Both types of components make available the software written to work with Java and R respectively (including database access libraries such as JDBC). Finally it is possible to execute local command line applications (such as perl scripts) through the "Execute cmd-line" component but it is discouraged since it reduces the workflow's portability, instead those with previously written components are encouraged to write SOAP interfaces to them.

Taverna workflows are able to incorporate typical flow of control methods, such as iteration and conditional branching. Iteration is implicit and based on the difference between a single input and a list of inputs. Basically, if a component expecting a single input is presented a list of such inputs, it will run once for each element in the list, generating a list of each of its outputs. The results of iteration can be 're-gathered' by a component whose input port expects a list of inputs, rather than a single input. There are no global variables and components cannot communicate except via their ports.

Once a workflow has been designed it can be executed, perhaps first providing any initial workflow inputs (such as a set of FLOSS project names). After animated step-by-step execution the final outputs of the workflow are displayed. The Taverna application understands how to display different MIME types, such as text, xml node trees, PNG and SVG graphics. Final outputs can each be saved as separate files.

One excellent feature of the software is that during and after the execution the full set of intermediate input and output variables can be viewed for each component, excellent for debugging or verification. Further breakpoints can be set and intermediate inputs manually edited during debugging. There is also an XML status report for each component available and the full set of status, intermediate and final results can be saved as an XML file for archiving or sharing.

Encouragingly the system provides significant metadata facilities. Firstly, a workflow or component designer can provide metadata, both in unstructured text descriptions and using scientific ontologies based on RDF. It is therefore possible to specify that a particular output port will be a string limited to a set of defined identifiers, in the FLOSS case perhaps a standardized project name.

[4] `http://r-project.org`

Secondly, the system provides a unique identifier for the workflow and a unique identifier for each and every workflow execution. These identifiers can therefore be used in papers to point to a specific workflow and the specific execution used to produce the results in the paper. The identifiers are called LSIDs (Life Sciences IDentifiers) and follow an open format, standardized by the Object Management Group[5].

2.2 Collaboration on workflows: MyExperiment.org

The group that developed Taverna has also developed a social networking site called MyExperiment to encourage sharing of workflows. The site allows the creation of profiles for individual researchers and the upload of Taverna workflows with a choice of licenses available (including the encouraged Creative Commons share-alike license). Users can then tag the workflows with metadata for discovery, can comment on the workflow and if the workflow is later used as a sub-workflow in another uploaded workflow the workflow receives a citation. Figure 4 shows the basic features of MyExperiment.org. The biology research focus of the community is clear from the tags, but it possible to create Groups of researchers and to restrict one's searches to that Group; it is anticipated that a FLOSS research group will have formed by the time of the demonstration. If the Group segmentation is not sufficient the entire MyExperiment.org application is open source and could be adapted specifically for the FLOSS research community.

3 Demonstration Proposal

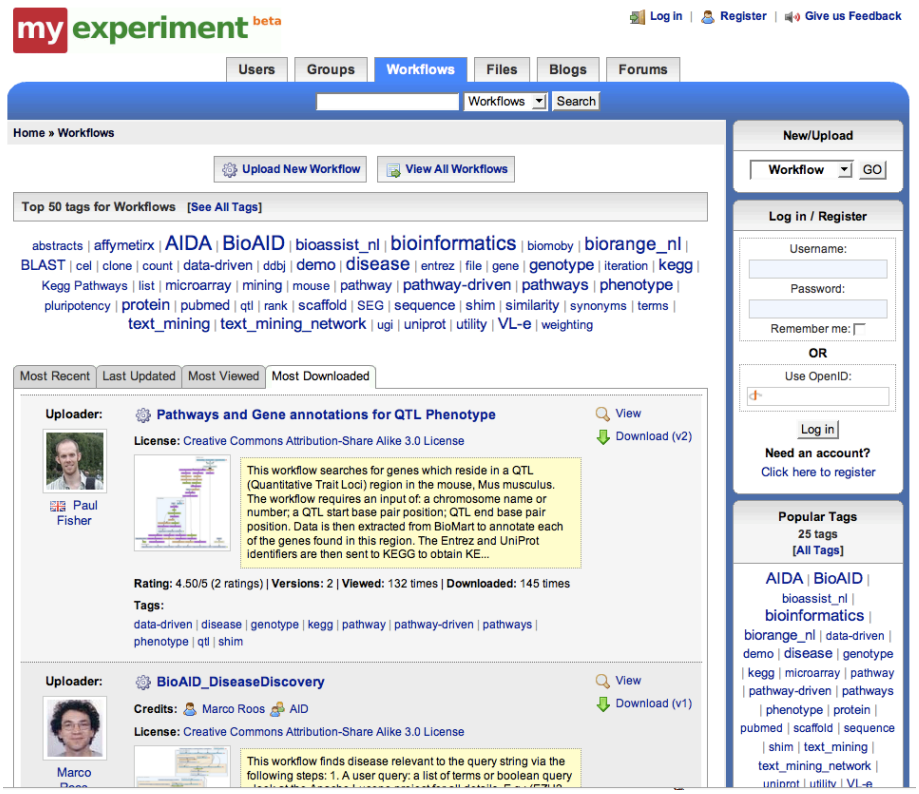
The demonstration proposed for the IFIP 2.13 conference in 2008 can be sized to fit an available time-slot and interest, but it would include the following elements:

- introduction to available FLOSS datasets
- introduction to scientific workflow concepts
- demonstration of the Taverna workflows in action
- live building of a simple workflow, using SOAP access to FLOSSMole and local components
- Demonstration of publishing that simple workflow on MyExperiment.org
- Discussion of how one would reference a workflow, and a specific workflow run, in a publication

Attendees would be provided with links to materials, including the sample workflows, so that they can work through the examples, and utilize the workflows, in their own research. Attendees will learn enough to be able to further explore the tools on their own and will be introduced to the library of existing components which will interface with existing FLOSS datasets.

[5] `http://www.omg.org/cgi-bin/doc?dtc/04-05-01`

Fig. 4. Screenshot from MyExperiment.org showing basic features: profiles, workflows and tags



3.1 Example workflows for demonstration

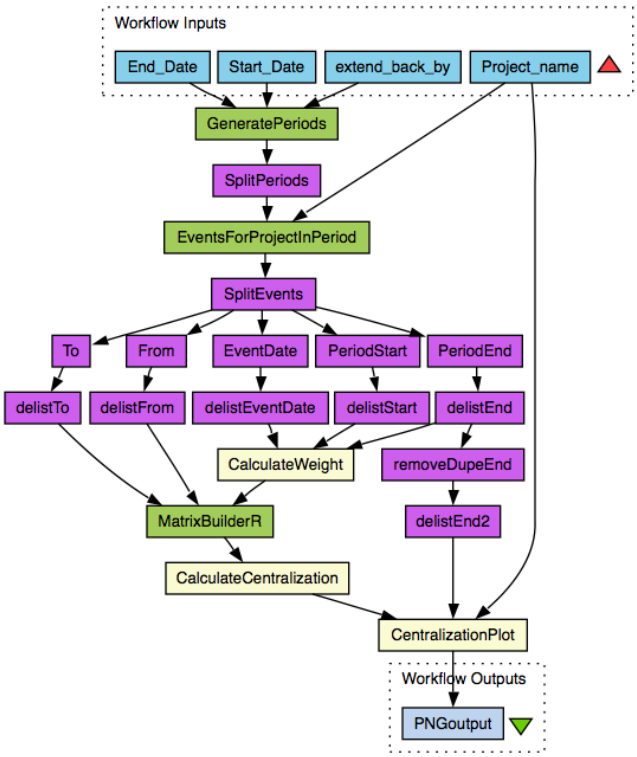
The authors have received NSF funding (Grant Number 0708767) to further their FLOSSmole project and are working to replicate—with Taverna workflows—a small number of studies from the research literature on FLOSS. These studies draw on large data sets (FLOSSMole, CVSanalY and the Notre Dame dumps) and will assist in the prototyping of SOAP access components and a library of reusable local components for the use of the research community. The studies currently candidates for replication include [12–16].

It is anticipated that these workflows will have been demonstrated at the NSF funded workshop "Free/Open Source Software Repositories and Research Infrastructures" hosted at UC Irvine in February 2008, so while they are not available at the time of this submission, they will be available for demonstration at the IFIP 2.13 conference in September 2008. What is available now, and

presented below, is a workflow prepared for a companion scientific paper [17], also submitted to the 2008 IFIP 2.13 conference.

The workflow draws on FLOSSmole data to produce a time-series graph of social network centralization through a project's lifetime, based on evidence from project communications. Figure 5 shows the workflow graphic saved directly from Taverna. Workflow inputs are boxed at the top, and the final graphical output boxed at the bottom. There are three types of components used:

Fig. 5. An example Taverna workflow for analysis of FLOSS communications social networks over time, saved directly from the Taverna interface

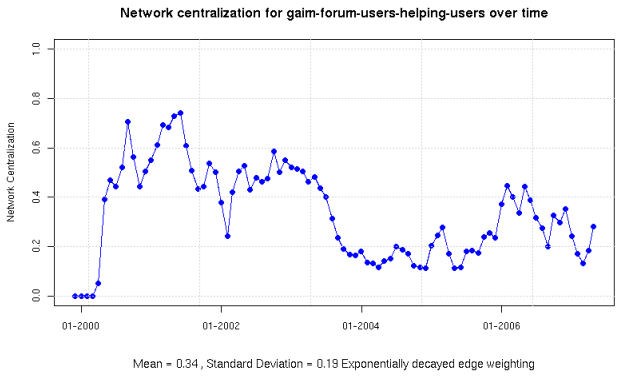


1. WebServices accessed via SOAP (GetPeriods, EventsForProjectInPeriod, MatrixBuilderR) (colored green on-screen)
2. Rshell components (CalculateWeight, CalculateCentralization and CentalizationPlot) (colored beige onscreen)
3. Local components used for spliting xml results and managing iteration (colored purple on-screen)[6]

[6] The awkwardness of the splitting, de-listing and re-listing elements in this diagram reflect our learning curve with Taverna, future work will rationalize these steps.

The basic flow is as follows. The user specifies a project and a time-period (say 1999 to 2006) and the workflow uses GeneratePeriods to get a set of periods[7] Then for each period the events (dated from-to relations in project communications) are accessed from FLOSSmole by EventsForProjectInPeriod, which returns an XML document which is split to individual events. Then the elements of each event are passed to CalulateWeight which performs a recency based exponential decay (so that less recent events are lower weighted) and the results passed to MatrixBuilderR. This component expects a list (from-to-weight) and so 're-gathers' the individual events for the period, returning an aggregated socio-matrix in a format that the sna library for *R* can understand. This matrix is then passed to *R* running in a local *Rserve* instance to calculate network centralization and the result passed, along with the appropriate end-date for the period to CentralizationPlot. This also expects list inputs and so re-gathers all the periods and produces a time-series graph and summary measures as shown in Figure 6. The workflow and the workflow execution xml files that produced this diagram are available in the FLOSSmole SVN[8]. The workflow execution run contains all the intermediate inputs and outputs and records of the number of iterations at each level.

Fig. 6. A sample output from the workflow above



The only component that directly accesses the FLOSSmole database is EventsForProjectInPeriod (this means that GetPeriods and MatrixBuilderR could have been implemented as local Beanshell componants, but we already had logic for these in our previously written Rails application, so simply wrapped them for SOAP access).

Preparing this workflow further convinced us of the usefulness of the workflow approach and of moving as much logic into it as possible. Firstly the availability of the inputs and outputs is ideal for debugging. Secondly the important

[7] GeneratePeriods can create overlapping periods when extendBackBy is set above zero. This is used for a sliding window, and interacts with the recency-based weighting to reduce double counting. See [17] for details.

[8] `http://ossmole.svn.sf.net/viewvc/ossmole/taverna-workflows/sna/`

step of entity resolution, by which we mean deciding which identifiers (email addresses, realnames, sourceforge user_ids) belong to the same person, currently occurs on the server as the data is imported to FLOSSmole's FLOSSEvent-Brower database. This hides the logic from the workflow and limits future users to the choices made on the server. Ideally the server would provide as raw data as possible and steps like entity resolution would be available as sub-workflows, so that others can alter the choices made and see the effect. Such sub-workflows are an excellent example of components that could be available in a library for community use.

4 Tasks ahead

The combination of growing large-scale public data sets and workflow tools such as Taverna and MyExperiment.org present a great opportunity for the eResearch on FLOSS and its development. There are, of course, issues to be worked through, including:

- building common interfaces to public datasets (eg SOAP/REST access)
- creating ontologies for naming parts of datasets, such as project and developer identifiers, venue types (user communication venues (lists, forums) vs developer venues (lists, forums, trackers)), while recognizing that these can change over time. RDF data formats will assist here.
- incorporating metadata data gathered about projects, such as their patch submission procedures and the location of their repositories at different times.
- incorporating specifically social science data, such as content analytic schemas and marked up data [18, 19], as well as interview recordings or transcripts and participant observation field notes (subject, of course, to informed consent and appropriate human subjects review).

Finally if this effort is to prove successful there must be encouragement by the community for researchers to get involved. In the first instance this means ensuring easy access to data sets as well as demonstrations such as the one proposed here. However the onus also falls on editors and reviewers to shape the research community's practices. For example, in order to discourage wasted effort and inconsistent data collection efforts, where appropriate and where the data is already publicly available, editors and reviewers should insist that the analysis in papers submissions be based on that data, even if it means re-running the analysis. Further reviewers should feel empowered to request the complete workflow, including statistical analysis, as part of their reviewing process.

These suggestions are, it must be made clear, separate to the question debated at last year's IFIP 2.13 conference, of whether authors should, as a condition of publishing on FLOSS topics, be required to make their collected data and analyses publicly available—workflows provided to reviewers could be done so with the expectation of confidentiality.

At the time this paper was prepared there were two workshops planned for February 2008 which will have advanced both data and collaboration in this domain. "Research Friendly" to be held Europe by the FLOSSmetrics project at FOSSEM, and the "NSF FOSS Repository and Research Infrastructures Workshop" hosted at UC Irvine. The presentation will draw on the outcomes of these workshops.

5 Conclusion

There are clearly trade-offs in standardizing on analysis technologies. For one there is a substantial store of experience and skills with individual researcher's tools of choice. Many research groups have talented programmers on staff who have built innovative in-house systems which work well for the problems that group has addressed. Standardization promotes collaboration but also asks research groups to move towards the standard tools, in order to benefit from the work of the collaborators. We have enjoyed working with Taverna, but encourage other FLOSS researchers to share their experience with other tools. One of the benefits of the modularity encouraged by the tools is that components can be relatively easily moved between software packages.

While standardization has some costs, the benefits of the collaboration it supports aren't limited to working with other research groups. In fact many groups have also had the experience of losing their main programmer, perhaps to graduation, and facing a lack of knowledge about their own systems. Indeed while it is commonly acknowledged that any form of collaboration between people can benefit from standardization, it is also true that programmers returning to their own work months later can also benefit from standardized approaches, enabling one to quickly build on one's own earlier work (as well as remember the exact choices one made).

eResearch presents a significant opportunity for research on FLOSS development. This paper outlines what is meant by a call for a move towards eResearch techniques and describes tools and ongoing work to kickstart that process. Finally it proposes a demonstration session for the IFIP 2.13 conference in 2008, which should make these ideas and tools concrete to the FLOSS research community.

References

[1] G. M. Olson, D. E. Atkins, R. Clauer, T.A. Finholt, F. Jahanian, T. I. Killeen, A. Prakash, and T. Weymouth. The upper atmospheric research collaboratory (uarc). *ACM Interactions*, 5(4):48–55, 1998.

[2] Daniel Atkins. Report of the National Science Foundation blue-ribbon advisory panel on cyberinfrastructure, 2003. URL `http://www.nsf.gov/od/oci/reports/toc.jsp`.

[3] F. Berman and H. Brady. Final report: NSF SBE-CISE workshop on cyberinfrastructure and the social sciences, 2005. Available at www.sdsc.edu/sbe/.

[4] NSF Cyberinfrastructure Council. Cyberinfrastructure vision for 21st century discovery, 2007. URL `http://netstats.ucar.edu/cyrdas/report/cyrdas_report_final.pdf`. NSF Report 0728.

[5] Bojan Cukic. The promise of public software engineering data repositories. *IEEE Software*, 22(6):20–22, 2005.

[6] J. Sayyad Shirabad and T.J. Menzies. The PROMISE Repository of Software Engineering Databases. School of Information Technology and Engineering, University of Ottawa, Canada, 2005. URL `http://promise.site.uottawa.ca/SERepository`.

[7] Megan Conklin, James Howison, and Kevin Crowston. Collaboration using ossmole: A repository of floss data and analyses. In *In Proceedings of the Workshop on Mining Software Repositories (MSR 2005) at the 27th International Conference on Software Engineering (ICSE2005)*, St. Louis, Missouri, USA., 2005.

[8] James Howison, Megan Conklin, and Kevin Crowston. Flossmole: A collaborative repository for floss research data and analysis. *International Journal of Information Technology and Web Engineering*, 1(3):17–26, 2006.

[9] James Howison, Megan Conklin, and Kevin Crowston. OSSmole: A collaborative repository for FLOSS research data and analyses. In *Proc. of 1st International Conference on Open Source Software*, Genova, Italy, 2005. URL `http://oss2005.case.unibz.it/`.

[10] Ionannis Antoniades, Ioannis Samoladas, Sulayman K. Sowe, Gregorio Robles, Stefan Koch, Ksenia Fraczek, and Anis Hadzisalihovic. D1.1 study of available tools. EU Framework deliverable, FLOSSmetrics, 2007. URL `http://flossmetrics.org/sections/deliverables/docs/deliverables/WP1/D1.1-Study_of_Available_Tools.pdf`.

[11] Audris Mockus. Large-scale code reuse in open source software. In *FLOSS '07: Proceedings of the First International Workshop on Emerging Trends in FLOSS Research and Development (FLOSS'07: ICSE Workshops 2007)*, page 7, Washington, DC, USA, 2007. IEEE Computer Society. ISBN 0-7695-2961-5. doi: http://dx.doi.org/10.1109/FLOSS.2007.10.

[12] James Howison, Keisuke Inoue, and Kevin Crowston. Social dynamics of free and open source team communications. In E. Damiani, B. Fitzgerald, W. Scacchi, and M. Scotto, editors, *Proceedings of the IFIP 2nd International Conference on Open Source Software (Lake Como, Italy)*, volume 203/2006 of *IFIP International Federation for Information Processing*, pages 319–330. Springer, Boston, USA, June 2006. URL `http://floss.syr.edu/publications/howison_dynamic_sna_intoss_ifip_short.pdf`.

[13] G. Robles, J. J. Amor, J. M. González-Barahona, and I. Herraiz. Evolution and growth in large libre software projects. In *The 8th International Workshop on Principles of Software Evolution*, Lisbon, Portugal, 2005.

[14] Scott Christley and Greg Madey. Global and temporal analysis of social positions at sourceforge.net. In *The Third International Conference on Open Source Systems (OSS 2007), IFIP WG 2.13*, Limerick, Ireland, June 2007.

[15] Megan Conklin. Do the rich get richer? The impact of power laws on open source development projects. In *Proceedings of Open Source 2004 (OSCON)*, Portland, Oregon, 2004. URL `http://www.elon.edu/facstaff/mconklin/pubs/oscon_revised.pdf`.

[16] Charles M. Schweik and Robert English. Tragedy of the foss commons? investigating the institutional designs of free/libre and open source software projects. *First Monday*, 12(2), 2007. URL `http://firstmonday.org/issues/issue12_2/schweik/index.html`.

[17] Andrea Wiggins, James Howison, and Kevin Crowston. Social dynamics of floss team communication across channels. In *Submitted to Fourth International Conference on Open Source Software (IFIP 2.13)*, 2008.

[18] Georg von Krogh, Sebastian Spaeth, and Karim R. Lakhani. Community, joining, and specialization in open source software innovation: a case study. *Research Policy*, 32(7):1217–1241, 2003. URL `http://ideas.repec.org/a/eee/respol/v32y2003i7p1217-1241.html`.

[19] Robert Heckman, Kevin Crowston, U. Yeliz Eseryel, James Howison, Eileen Allen, and Qing Li. Emergent decision-making practices in free/libre open source software (FLOSS) development teams. In Joseph Feller, Brian Fitzgerald, Walt Scacchi, and A Sillitti, editors, *Open Source Development, Adoption and Innovation*, volume 234 of *IFIP International Federation for Information Processing*, pages 71–84. Springer, Boston, USA, 2007.