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 introduces eResearch workflow tools as a model for the research community studying free and open source software and its development. 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 suitable research data sets the paper introduces a class of tools known as scientific workflow frameworks, focusing on one—Taverna—and introducing its features. To further explain the tool a complete workflow used for original research on FLOSS is described. Finally the paper considers the trade-offs inherent in these tools.[1]

eResearch refers to a set of scientific practices and technologies, sometimes called eScience or Cyberinfrastructure, 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) and the NSF has produced a series of reports on the topic (eg [4]). The hallmarks of eResearch are: a) broad community-level collaborations between distributed scientists, b) large-scale broadly available data sets c) shared computational analysis tools and workflows, and d) 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. Given that we study highly effective distributed collaborators and collaborative technologies in the FLOSS community, we have good understanding of solutions to the challenges faced by such distributed collaborations. Figure 1 shows an envisioned workflow repository which allows the discovery, replication, extension and publication of research workflows, drawing on shared components. It is time to build further towards eResearch.

1 Current FLOSS research practices

A series of papers has examined the current research practices in the investigation of FLOSS and its development [1, 2][2]. In general this research has been, and continues

[1] This work is supported by NSF grant #0708437. A longer version of this paper is available at http://floss.syr.edu/publications/HowisonTavernaDemoIFIP.pdf

[2] See also the report of the FOSSRRI workshop at http://fossrri.rotterdam.ics.uci.edu/ and the Research Room @ FOSSDEM: http://libresoft.es/Activities/Research_activities/fosdem2008

Please use the following format when citing this chapter:

Howison, J., Wiggins, A. and Crowston, K., 2008, in IFIP International Federation for Information Processing, Volume 275; *Open Source Development, Communities and Quality*; Barbara Russo, Ernesto Damiani, Scott Hissam, Björn Lundell, Giancarlo Succi; (Boston: Springer), pp. 405–411.

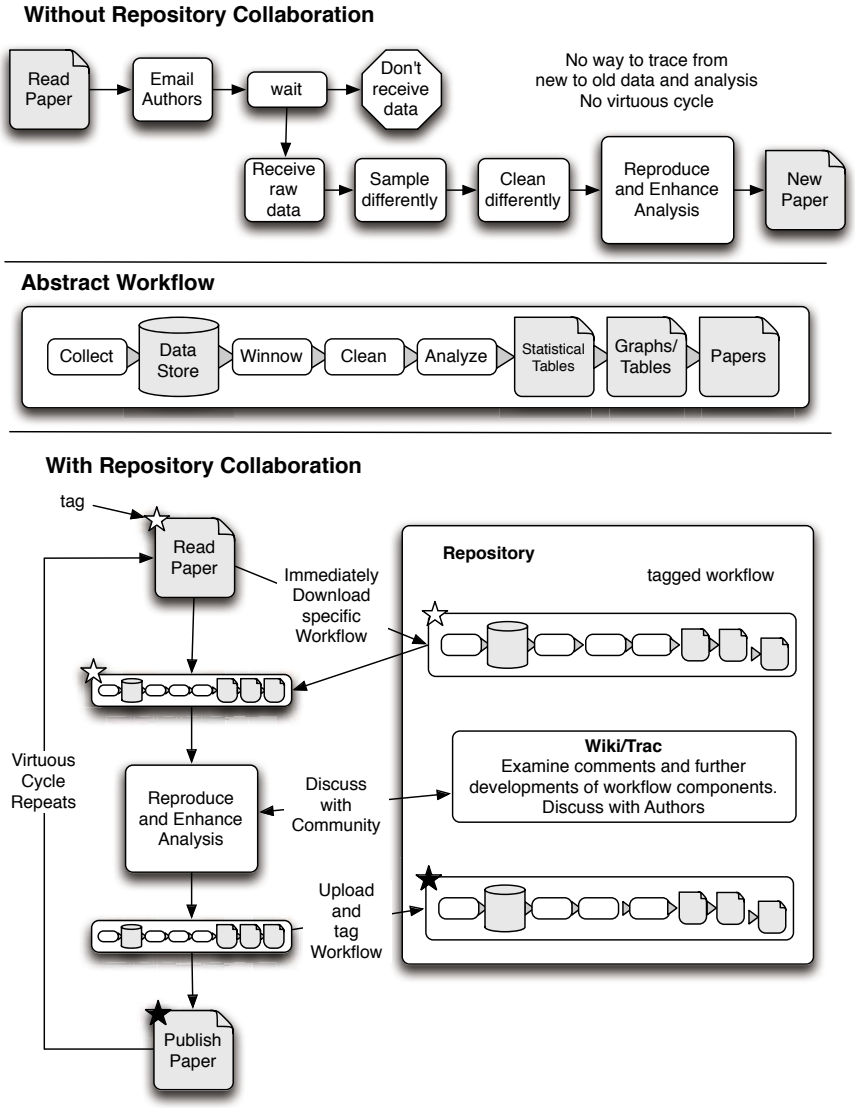
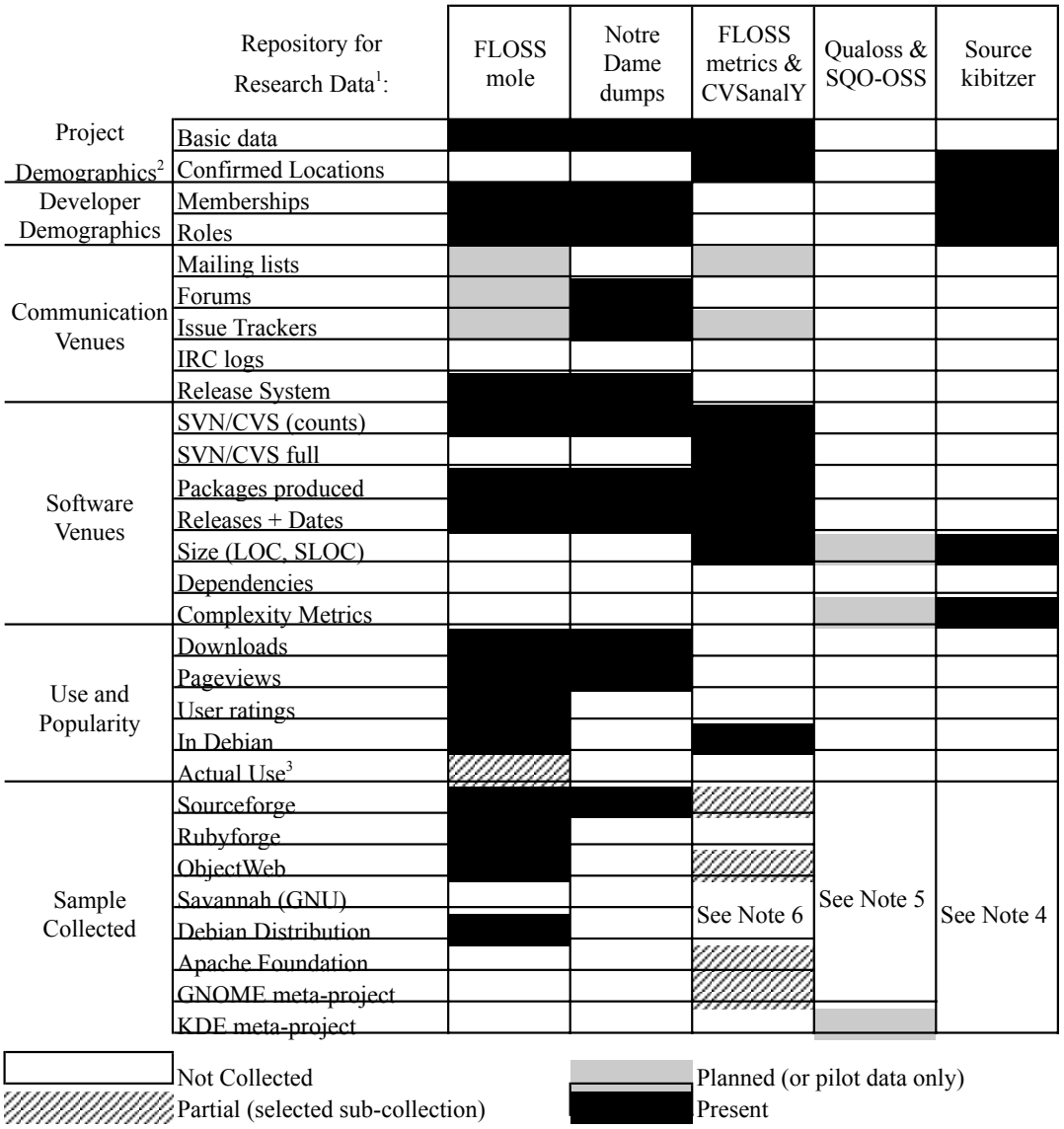


Fig. 1. Envisioned improvements in FLOSS research practices

to be, undertaken by separate groups and has involved substantial re-collection of very similar data, usually through spidering Sourceforge or similar sites.

In the past three years significant progress has been made towards shared datasets held in what have been called Repositories of Repositories [1]. At the IFIP 2.13 conference in 2007 a workshop was held for research based on public databases, including FLOSSMole and CVSanalY. In addition, the Notre Dame Sourceforge database dumps are available under an academic sub-license. 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, and points out gaps in data availability. Of course not all researchers use these databases, with many continuing to spider their own data sets, especially outside of such communities as IFIP 2.13 and the Mining Software Repositories workshops.



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. 2. Table showing repositories available for FLOSS research

While progress has been made in data availability there is little sharing of analyses, including components that 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 [3] and [5]) have merely placed such bespoke tools on project websites. This paper now turns to describe tools that can move research on FLOSS towards increased collaboration and better research results.

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 workflows.

Scientific workflow tools such as Taverna and Kepler support high-level programming which binds together data sources and analysis steps. 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 3, below) and is usually stored in a single XML file. As with most programming environments, 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 high-level composition also promotes modularity in analysis development, believed to lead to easier collaboration and higher quality products.

Taverna The proposed demonstration focuses on Taverna and uses workflows developed to address research questions about FLOSS development. Taverna is instantiated as a stand-alone desktop application, written in Java and therefore cross-platform. Workflows can also be run via a 'headless' server application.

The application has two main interface modes: one for the design of a workflow and one for its execution. The design mode provides a list of available components, the workflow definition, and an automatically rendered diagram of the workflow. 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.

Taverna supports workflows with both remote and local components. Remote services can be gathered ('scavenged' in Taverna parlance) by entering URLs including Web Services Description Language (WSDL), or from other workflows. It is anticipated that the development of remote services in the FLOSS domain will utilize the WSDL/SOAP combination, standard in many server technologies. Taverna parses the WSDL description file to make multiple components available, each with named input and output ports.

Local services include a library of components dealing with standard operations such as file IO, string and list manipulation. Customized local components can be written in Java, via a scripting syntax called Beanshell, and in the statistical package *R*. Data fed into a component's 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 script. Workflows are able to incorporate typical flow of control methods, such as iteration and conditional branching. There are no global variables and components do not communicate except via their ports.

A designed workflow is executed by first providing any initial workflow inputs (such as a set of FLOSS project names or sampling criteria). An animated step-by-step process monitor summarizes analysis progress until the final outputs of the workflow are displayed. 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, significantly improving workflow debugging and verification. An XML status report is available for each component, and the full set of status, intermediate and final results can be saved as an XML file for archiving or sharing.

Taverna provides significant metadata facilities. Firstly, a workflow or component designer can provide metadata about the workflow or individual ports, both in unstructured text descriptions and using scientific ontologies based on RDF/OWL. Secondly, the system provides a unique identifier for the workflow and a unique identifier for each and every workflow execution. These identifiers are standardized by the Object Management Group and can be used in papers to point to a specific workflow as well as the specific execution used to produce the results in the paper.

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, groups (such as our FLOSS group), and the upload of Taverna workflows. Users can tag their contributed workflows with metadata for discovery and can download, rate, and comment on workflows. If the workflow is later used as a sub-workflow, a citation is displayed on the site.

Example workflow The authors are working to replicate a small number of FLOSS studies with Taverna workflows. These studies draw on large federated data sets (FLOSSMole, CVSanalY and the Notre Dame dumps) and will assist in the prototyping of SOAP access and a library of reusable local components. The discussion below presents a workflow prepared for a companion scientific paper [6], also published in these proceedings.

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 3 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: 1) WebServices accessed via SOAP (eg GetPeriods) 2) Rshell components (eg CalculateWeight) and 3) Local components used for splitting XML results and managing iteration.

The basic flow is as follows. The user specifies a project and a start and end date, for which GeneratePeriods provides a set of overlapping periods. Then for each period, the events (dated from-to relations in project communications) are accessed from FLOSSmole by EventsForProjectInPeriod, which returns an XML document of the events. The XML is split into individual events, used by CalulateWeight to calculate a recency based edge weight, such that less recent events are lower weighted.

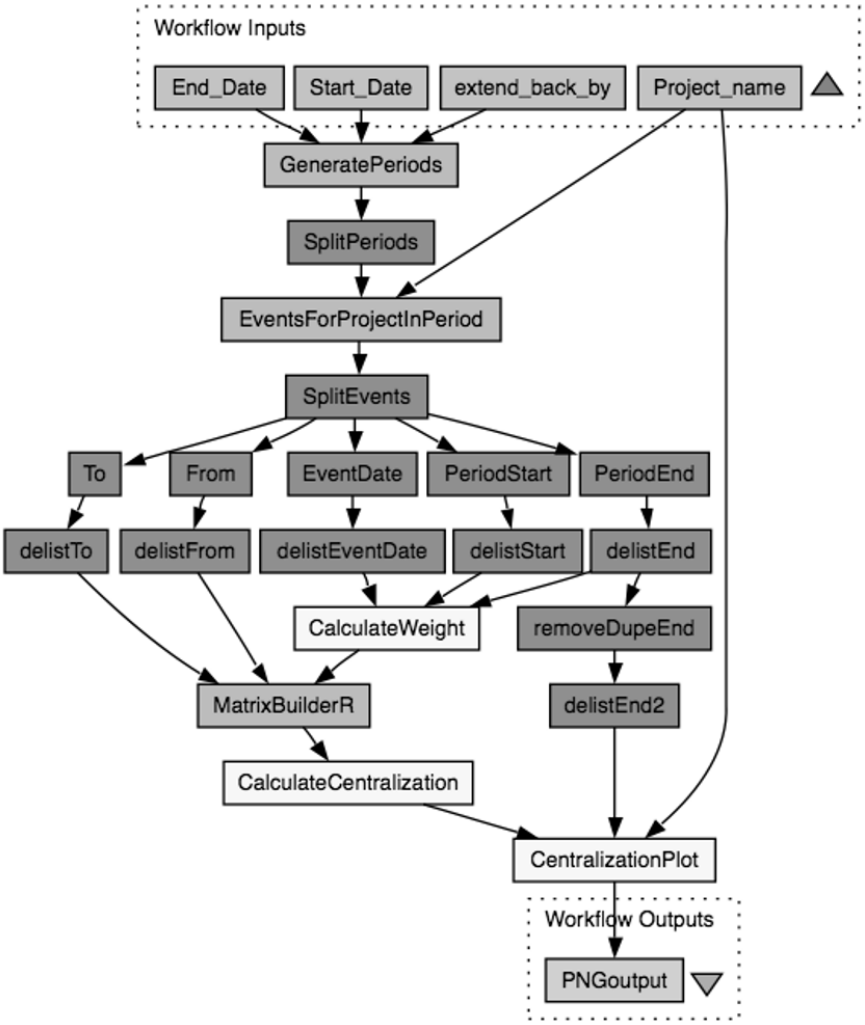


Fig. 3. An example Taverna workflow for analysis of FLOSS communications social networks over time, saved directly from the Taverna interface

The MatrixBuilderR component uses the dates and weights to generate an aggregated socio-matrix. A local *Rserve* instance uses the matrix to calculate network centralization, which is passed with an associated date to the CentralizationPlot component to produce a time-series graph and summary measures. The workflow and the workflow execution XML files that produced this diagram are available in the FLOSSmole SVN (see [6]).

3 Conclusion

The combination of growing large-scale public data sets and workflow tools such as Taverna and MyExperiment.org present a great opportunity for eResearch on FLOSS and its development. There are, of course, issues to be worked through, including a) building common interfaces to public datasets, b) creating ontologies for naming parts

of datasets, such as project and developer identifiers, c) incorporating metadata gathered about projects, such as their patch submission procedures and the location of their repositories at different times and d) incorporating social science data, such as content analytic schemas. While data sets and demonstrations are vital to encouraging researcher involvement, editors and reviewers should feel empowered to request complete workflows and insist that papers draw on available datasets, where available.

There are clearly trade-offs in standardizing on analysis technologies. For example, there is a substantial store of experience and skills with individual researcher's tools of choice. Standardization promotes collaboration but also asks research groups to move towards the standard tools, in order to benefit from the work of the collaborators. While standardization has some costs, the benefits of the collaboration it supports aren't limited to working with other research groups. Many groups have also had the experience of losing their main programmer, subsequently facing a lack of knowledge about their own systems. Indeed while it is commonly acknowledged that any form of collaboration 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.

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 to make these ideas and tools concrete to the FLOSS research community.

References

[1] Antoniades I, Samoladas I, Sowe SK, Robles G, Koch S, Fraczek K, Hadzisalihovic A (2007) D1.1 study of available tools. EU Framework deliverable, FLOSSmetrics, URL http://flossmetrics.org/sections/deliverables/docs/deliverables/WP1/D1.1-Study_of_Available_Tools.pdf

[2] Howison J, Conklin M, Crowston K (2006) FLOSSmole: A collaborative repository for FLOSS research data and analysis. International Journal of Information Technology and Web Engineering 1(3):17–26

[3] Howison J, Inoue K, Crowston K (2006) Social dynamics of free and open source team communications. In: Damiani E, Fitzgerald B, Scacchi W, Scotto M (eds) Proceedings of the IFIP 2nd International Conference on Open Source Software (Lake Como, Italy), IFIP International Federation for Information Processing, vol 203/2006, Springer, Boston, USA, pp 319–330, URL http://floss.syr.edu/publications/howison_dynamic_sna_intoss_ifip_short.pdf

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

[5] Robles G, Amor JJ, González-Barahona JM, Herraiz I (2005) Evolution and growth in large libre software projects. In: The 8th International Workshop on Principles of Software Evolution, Lisbon, Portugal

[6] Wiggins A, Howison J, Crowston K (2008) Social dynamics of FLOSS team communication across channels. In: Proceedings of the Fourth International Conference on Open Source Software (IFIP 2.13), Milan, Italy