eScience for Free/Libre Open Source Software Researchers

Kevin Crowston, Andrea Wiggins, & James Howison
crowston@syr.edu, awiggins@syr.edu, jhowison@syr.edu

Syracuse University School of Information Studies
343 Hinds Hall, Syracuse, NY 13244
Phone: +1 (315) 443-1676
Fax: +1 (866) 265-7407

Abstract. This paper presents a case study of the application of eScience tools and practices for the social science research community studying Free/Libre Open Source Software (FLOSS) development practices. We first describe what we mean by eScience and introduce research on FLOSS to motivate the need for eScience tools and approaches. We then describe our initial efforts to introduce eScience tools for FLOSS research, potential obstacles, and how the use of such tools might affect the practice of research in this field.

Keywords: eScience, open source software, social science research, cyberinfrastructure

1 Introduction

eScience is the use of information and communications technologies (ICT) to support scientific work. Nentwich (2003) defines eScience (using the term CyberScience) more specifically as "scholarly and scientific research activities in the virtual space generated by the networked computers and by advanced information and communication technologies" (Nentwich 2003, p. 22). eScience is also known as eResearch, CyberScience or Science 2.0, and the supporting systems as cyber-infrastructure or scientific collaboratories. The different terms are popular in different settings, eScience being common in Europe, while the US National Science Foundation (NSF) typically refers to cyber-infrastructure. Because there are many types of ICT and many kinds of scientific activity to which they can be applied, eScience encompasses a broad range of distinct cyber-infrastructure applications. For many, cyber-infrastructure means high performance computing, e.g., grid computing to support analyses of large volumes of data

and simulations, for many the third approach to science, between theory and experiment. But cyber-infrastructure also includes Internet-enabled applications to connect scientists to a variety of resources: data, knowledge and other researchers.

Data might come from instruments directly connected to the Internet, a shared instrument, or from structured data repositories in a community data system. *Knowledge* might be captured in digital libraries of journal articles or, increasingly, of article preprints. Connections between data and knowledge can also be made explicit; a simple step is to link publications to data sources and vice versa. More interestingly, scientific workflow tools can be used to capture analysis steps explicitly in an executable format, enabling reuse and sharing of analyses. Increasingly, eScience applications are built using semantic web technologies that enable automated reasoning about scientific knowledge. Finally, because science is not a solitary pursuit, eScience applications can also include groupware to connect researchers to *other researchers* and thus support scientific collaboration. These applications can range from simple email lists connecting collaborators, to digital libraries of various scopes, to newer collaborative applications such as wikis, shared document editors or semantic web reasoners. These tools can support virtual collaborations of different scales and degrees of formality. For example, discussion boards linked to papers in preprint archives have been proposed to augment or even replace some of the functions of conventional peer review (Nentwich, 2003). To the above list of technologies, we must add the necessary social institutions needed to make the technology successful, a point that we will discuss in more detail below.

As with the applications of ICT to other kinds of work, eScience is presented as having substantial promise to reshape and enhance the way science is done. For example, a report by the US National Science Foundation (NSF) Cyberinfrastructure Council, *Cyberinfrastructure*

Vision for 21st Century Discovery, calls a "comprehensive cyberinfrastructure essential to 21st century advances in science and engineering research and education", and NSF has backed this vision by the creation of an Office of Cyber-Infrastructure to fund eScience efforts. However, the reality of eScience implementation is more mixed. While eScience is extensively applied in some fields such as physics or biology, it is scarcely visible in others. This paper introduces the concepts behind eScience and then presents a case study of the application of these ideas to a particular research area, namely social science research on the development of Free/Libre Open Source Software (FLOSS). This paper has two purposes: to identify potential applications of eScience ideas to benefit the FLOSS research area and to use this case study of eScience in a particular field to reflect on issues that arise in the application of eScience in the social sciences more generally.

2 eScience for FLOSS research

In this section, we explore how eScience is and could be applied to the social science study of free/libre open source software (FLOSS) projects. FLOSS research presents a useful case study, as it is interdisciplinary, small enough to comprehensible but diverse enough to present a variety of issues, and has some nacent eScience applications. Our goal is to explore characteristics of FLOSS research to understand the extent and potential for the adoption of eScience tools and practices in this field, shedding light on the broader applicability of these ideas. We therefore start by sketching the state of existing research on FLOSS.

FLOSS is a broad term used to embrace software released under an “open source” license and often developed in a community-supported mode. Due to their size, success and influence, the Linux operating system and the Apache Web Server are the most well known FLOSS projects, but hundreds of others are in widespread use. Over the past ten years, FLOSS

has moved from a curiosity to mainstream, with a concurrent increase in the amount of research examining this phenomenon. Some of this research has focused on FLOSS as a new approach to innovation in the software industry, characterized by a rapid and reliable software development process. In addition, FLOSS has attracted great academic interest because it provides an accessible example of other phenomena of interest. For example, researchers have turned to FLOSS projects as examples of virtual work, as they are dynamic, self-organizing, and composed of distributed individuals working in loosely coupled teams. Effective FLOSS development teams somehow profit from the advantages and overcome the disadvantages of distributed work, and their emphasis on distributed work makes them useful as a research setting for isolating the implications of this organizational innovation. Traditional organizations have also taken note of these successes and have sought ways of leveraging FLOSS methods for their own distributed teams.

FLOSS as an object of study appears to be particularly apt for the application of eScience practices, most clearly in terms of data availability. FLOSS teams create and retain public archives as a record of their activities. These archives provide researchers unparalleled access to data on the processes of these teams. As well, the source code of the systems is always available, often for many releases and even change-by-change in source code control systems, such as CVS. However, the ready availability of primary data is a misleading indicator of the nature of FLOSS research. Precisely because data are by-products, they are rarely in a form useful for researchers. Instead data are embedded in HTML pages, CVS log files, or text-only mailing list archives. Furthermore, because FLOSS projects are hosted in a variety of "forges" (of which Sourceforge is the largest) or individual websites with differing structures, these problems are multiplied. Research projects expend significant energy collecting and re-

structuring these data sources for their research. FLOSS research to date has often involved redundant data collection by independent research groups, usually through spidering Sourceforge or similar FLOSS development sites (Howison et al., 2005, Antoniades et al., 2007, Conklin et al., 2005, Howison et al., 2006a).

To support FLOSS research and to help the diverse collection of researchers interested in FLOSS mature into a research community, we envision a shared cyber-infrastructure that will facilitate access to data, analyses, papers and other researchers. This infrastructure will include both a technological base as well as a set of enabling social mechanisms. Initial building blocks of this infrastructure already exist but considerable additional work is needed, as we discuss below. In the remainder of this section, we discuss possible applications of eScience to FLOSS research, before turning to a more detailed example in the following section. To make the overview discussion more concrete, we will consider the example of a researcher interested in testing Conway's law, which states that the structure of product mirrors the social structure of the group that develops it (Conley, 2008, Conway, 1968, Herbsleb and Grinter, 1999).

2.1 Data

To test our example proposition, we first need data for team structures and the structure of their software products over a period of time, ideally for a number of different teams operating under different conditions. As discussed above, FLOSS research is facilitated by the fact that much of the relevant data is "born digital", and with the growing use of ICT, an increasing volume of such "trace" data is available in many settings. Significant progress has been made in the past few years in creating shared data sets held in what have been called Repositories of Repositories (RoRs) (Antoniades et al., 2007), which provide raw data on FLOSS projects and the products they build, such as FLOSSMole (Howison et al., 2006a), the Notre Dame SourceForge

repository (http://www.nd.edu/~oss/Data/data.html) and CVSanalY (Robles et al., 2004). These repositories provide project-level data for thousands of projects, including project characteristics, forums, trackers, source code, and so on. Making all of the available data usable also requires efforts to create better metadata, such as data dictionaries, provenance information, and entity resolution for developers' online identifiers. A useful contribution could be made simply by cataloging the available data, e.g. (Howison et al., 2008).

A second challenge in finding data for the example study is developing a suitable sample of projects. As with any kind of positivist research, the key requirement is an explicit population of interest for the research question and a representative sample selection. Developing such a sample for FLOSS research can be fraught with complications. For example, many projects registered in a forge may not have released software, making them inappropriate for the example research question. A subtler problem is that code released may not be the product of a group effort, meaning that there is no group structure to compare to the code structure. More generally, researchers may want to test their propositions with groups of a particular type, e.g., distributed groups, or projects that have released mature code. Researchers would therefore benefit from accessing information such as sampling frames and previously studied samples of projects known to be comparable on different dimensions. A possible application would be a shared census of FLOSS projects to establish the universe of study, including descriptive data about projects, as well as ways to describe the sample used in any particular study.

2.2 Analyses

Given the data and an appropriate sample, a further challenge is raising the level of analysis from raw data to the concepts of interest to test the propositions. Again, connecting data and theory is a problem in any research, but is particularly problematic in the social sciences, where

concepts may lack precise operationalizations. As well, such connections are particularly difficult when dealing with trace data, which are the byproducts of behaviours rather than measures of theoretical constructs. In the example research problem, concepts of interest are social structure and software structure, which can be defined and operationalized in many ways, using social network analysis (e.g., (Crowston and Howison, 2005)) or formal team membership, geographical proximity, or person-to-person contacts (Cataldo et al., 2006). Software structure can be measured with an array of metrics, such as coupling and cohesion (Conley, 2008, Kemerer and Slaughter, 1999).

To connect our sample proposition to the existing literature, we would prefer where possible to reuse accepted conceptual definitions. Theoretical definitions and operationalizations of concepts are typically described in research publications, but even with full-text searching, identifying papers that have used a particular concept is difficult. Furthermore, it is often the case that published descriptions of data analysis are too terse to be easily replicated, a problem exacerbated by publication length restrictions. For example, data often require complex cleaning steps, which are rarely described to the level of detail required to duplicate the results. One possible improvement is sharing of intermediate research results, either as new data sets or as annotations on existing data. For example, email data might be coded for various theoretical concepts, e.g., those showing team structure and source code might be analyzed for code structure, complexity or to illuminate project interdependencies. The resulting datasets could themselves be shared, with appropriate metadata describing their provenance, and used as a starting point for further studies.

A more useful form for conceptual definitions is a computational representation of their operationalizations, which allows them to be tweaked, critiqued, and easily applied to new data.

Scientific workflow systems are one type of tool which support high-level programming that binds together data sources and analysis procedures. Known collectively as "scientific workflow tools", examples include Taverna (http://taverna.sourceforge.net) and Kepler (http://kepler-project.org). The software follows the same basic principles: steps in a workflow are performed by modular components with multiple input and output ports through which the components are linked. As with most programming environments, much of these tools' utility derives from their library of local and remote components, which can be applied to both quantitative and qualitative data. The workflow specification can be shared via email or posted on a website as an appendix to a publication. In this case, a researcher might retrieve a workflow that computes team structure from interactions on a developer email list, thus ensuring the use of the same definition of the concept as in prior work, or allowing a more precise comparison of existing and novel conceptualizations. In a later section of the paper, we will present a more detailed example of the use of workflow tools to illustrate these points.

2.3 Other researchers

Finally, our example researcher could benefit from interaction and collaboration with other FLOSS researchers to help them hone the research questions and approaches. The FLOSS research community already has several mailing lists on which researchers share news and occasional questions about data or research approaches. The lists associated with particular data repositories seem to have more substantive discussion, focused by the challenges of working with the data.

However, these are still relatively minor efforts, leaving great opportunity to enhance interaction among researchers. For example, the FLOSS research community has a preprint database with more than 200 articles and efforts are being made to improve the articles'

metadata to facilitate access, e.g., by cataloging the concepts examined in the articles. The repository could support discussion of posted articles, thus mirroring some of the functions of conference presentation or peer review, but over longer timescales with wider audiences. Sharing intermediate research results might facilitate identification of possible research collaborators so that rather than duplicating work, researchers might be able to draw on the specialized knowledge of others. More ambitiously, the community could develop shared information resources, such wikis or more structured knowledge bases of research findings.

3 Case Study: Replicating FLOSS research using cyber-infrastructure

In this section, we present a more detailed case study of the use of one class of cyber-infrastructure to support FLOSS research. As a proof of concept of the applicability of eScience ideas to FLOSS research, the authors (with Megan Conklin of Elon University) have received US National Science Foundation funding for a project that includes using data repositories and Taverna workflows to replicate five published studies. In this section, we discuss this replication effort in more detail. The selected studies for replication are shown in Table 1. These studies were chosen because they draw on the large data repositories described above, spanning a range of research questions and approaches amenable to automated analysis.

Our replication effort is expected to have several benefits. First and foremost, it provides evidence to the community of the applicability of the tools of eScience, thus promoting adoption. Second, by replicating the studies using shared data sources and analysis workflows, we can extend the original analyses, e.g., by testing alternate choices of variables used to determine project success or applying the analysis to additional data covering larger periods of time or more projects. We also found benefits from more precise operationalizations of the concepts in the papers. Finally, the effort will provide a set of local and SOAP components that

can be reused in other workflows.

Table 1. Studies chosen for replication.

Study	Description
Crowston & Scozzi, 2008	Applies competency rallying approach to predict success of projects based on various project factors.
Conklin, 2004	Examines if distribution of project sizes is consistent with a developers joining projects in a scale-free network.
Howison et al., 2006b	Examines dynamics of social networks of project communications over time.
Robles et al., 2005	Examines growth rate of software.
English and Schweik, 2007	Classifies FLOSS projects based on metrics for success versus abandonment and stage of project growth.

In the first part of this section, we discuss the experience of developing replication workflows before summarizing some lessons learned in the following section. At a high level, the steps in replication were fairly straightforward. We began by evaluating the data, methods, analysis, and findings reported in our selected papers. From this, we generated a specification of data requirements, drawing on data in the repositories, and a list of desired outputs, followed by an abstract workflow representing the expected analysis to get from data to outputs. In doing so, we consulted the original authors of the works that we were replicating for advice on their original data selection parameters and to clarify the operationalization of some constructs in the papers. Workflow components were then created or selected and linked together to achieve the operational specifications represented in the abstract workflow. The resulting workflow underwent iterative testing and development until it was fully functional. Our process for replicating research was collaborative, as the more complex workflows required the cooperative efforts of two individuals with complementary skills. When the workflow was considered

functionally complete, it was documented, shared with other researchers, and used to explore the analysis results. The ease with which we were able to collaboratively develop complex analyses demonstrates several of the benefits of using Taverna and the agile, iterative approach that we employed in our collaborative development efforts.

3.1 Building FLOSS workflows

We start by describing in more detail the development process, using the English & Schweick (2007) paper as an example. This paper used various FLOSS project metrics to classify projects according to stage of growth and success versus abandonment. The classification requires retrieving and consolidating data about each project (e.g., release history) and then sorting projects into categories based on this data. As mentioned above, the first step of the replication is identification of the data needs for the workflow, followed by development of an abstract workflow that captures the main steps in the analysis. We then translate the abstract processes into operational processes, building a workflow by directing inputs through modular component processors into outputs; the components can be RShell (running the R statistical package with an RServe instance), Beanshell (simplified Java scripting), command-line operations, SOAP web services or local Java shims for common operations. In the abstract workflow for the classification (Figure 1), the inputs are parameters to the workflow; project data are retrieved from the databases in the get_data_subworkflow component. The data are manipulated for analysis in the summarize_releases_subworkflow and the Concatenate_two_strings Java shim. The classification_subworkflow component represents the main analytical process of classifying projects based on the project data, and in this early stage of workflow development, the specific workflow outputs were not yet defined.

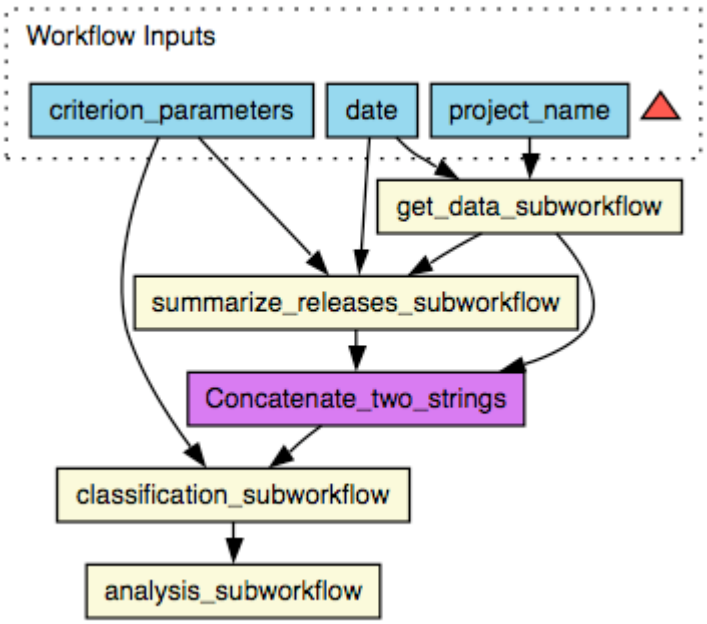

Figure 1: The collaboratively developed abstract workflow for the replication of the classification of FLOSS projects from (English and Schweik, 2007).

Development of the workflow components had an unanticipated benefit. As we worked through translating the classification criteria into a truth table to describe the classification subworkflow, we discovered that several cases were not described in the table of the published paper, likely because no projects with that mix of characteristics existed in the original sample. By applying the principle of inheritance to some of these missing cases, we were able to address the gaps in a manner consistent with the original classification, but one negative case remained somewhat ambiguous.

3.2 Using the workflows

After developing each workflow, we applied it to replication and exploration. To validate that a workflow correctly captured the original authors' intent, we ran it with a data sample appropriate for comparison to the original work. For some replications we had more data available due to the passage of time, and in others we had less due to the limitations of the

repository data sources. In these cases, we made the most reasonable comparison possible, seeking a similar distribution or proportion of classification outcomes to the original work. An example of our results are shown for the Conklin study in Figures 2 – 5 (Conklin, 2004). Conklin's analysis showed that the network of projects and developers exhibits a scale-free structure. As a comparison of the figures shows, the original and replicated distributions are very similar despite a marked increase in number of projects analyzed, from under 20,000 in the original work to over 65,000 in the replication analysis.

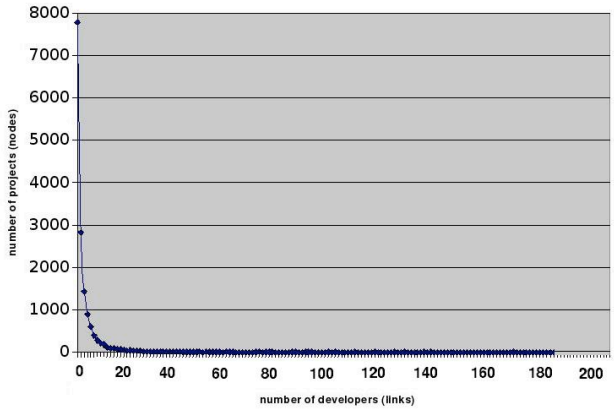

Figure 2

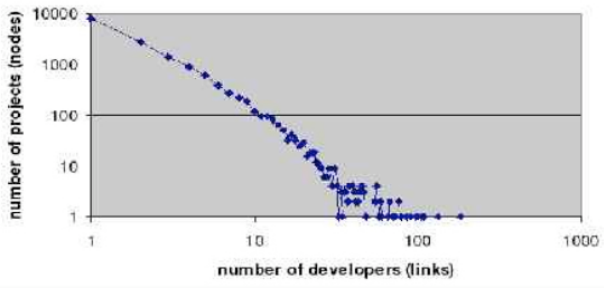

Figure 3

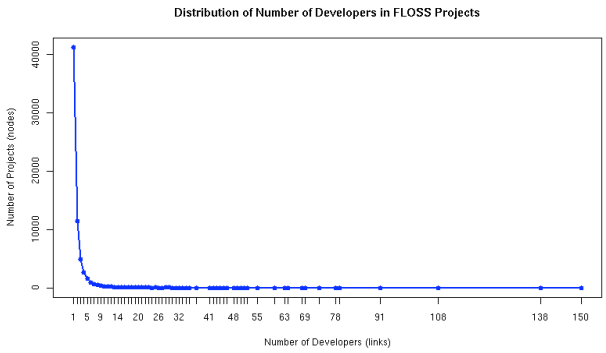

Figure 4

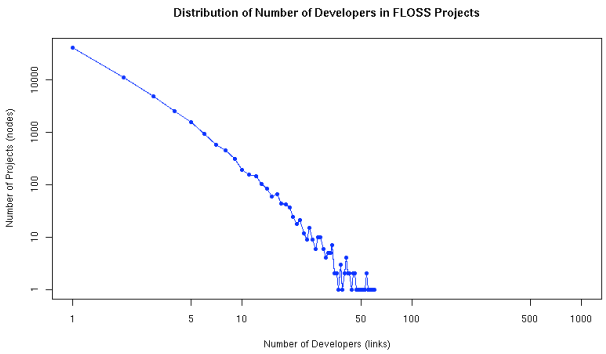

Figure 5

Figures 2, 3, 4, 5: our replication (Figures 4 & 5) mirrors the original findings (Figures 2 & 3, used with author's permission) from the Conklin presentation on the scale-free structure of FLOSS developer-project networks (Conklin, 2004).

Recreating prior research analyses also provided the opportunity to extend the original

work. For some workflows, we implemented the future work suggestions from the authors, as well as our own ideas for alternative processes. For example, developing two alternative approaches for calculating a particularly complex metric for a construct in the English & Schweick (2007) classification workflow was simple once the rest of the workflow design was in place. The immediate result of comparing the authors' original release rate measure to the one that they proposed as future work was striking: the tested values for a handful of projects known to have achieved successful software production shifted from a classification status affirming their success to an indeterminate status.

Workflows also facilitate sensitivity testing on critical thresholds or input values. For example, the Howison et al. (2006a) paper used a sliding time window with network data to create dynamic social networks. Implementing this analysis as a workflow made it easy to examine the effects of changing the window size (Figure 6). While exhaustive sensitivity testing remains to be completed, we expect to implement this analysis with a reusable sensitivity testing workflow to automatically analyze and compare combinatorial sets of parameter values. The use of a workflows makes the otherwise onerous task of sensitivity testing much more manageable for a complex analysis, allows the researchers to more fully explore their analysis parameters and thus enables a more rigorous validation of the analysis results.

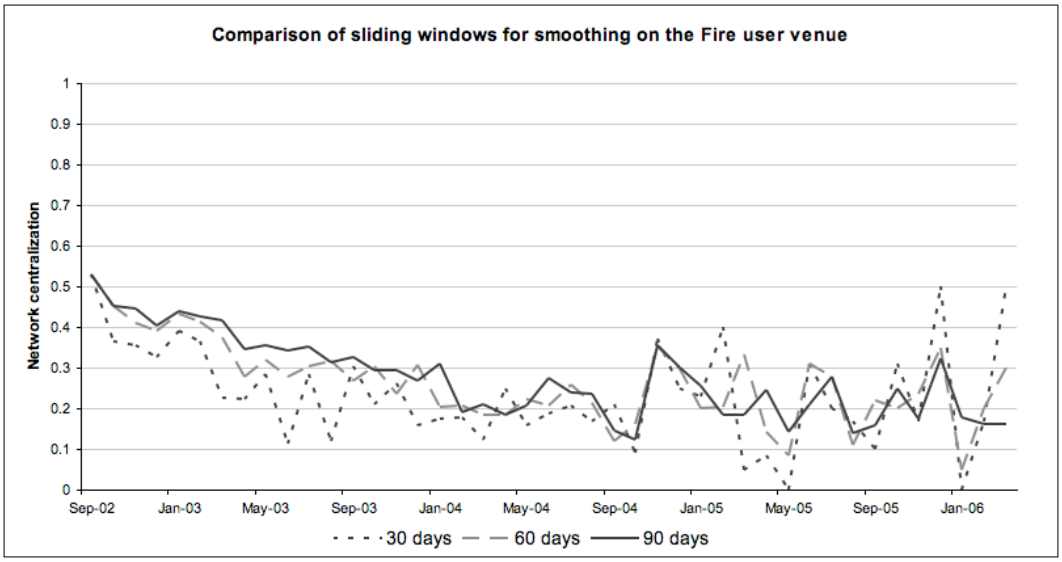

Figure 6: Workflow analysis tools allow easy comparison of analysis parameters, such as the length of a sliding window for smoothing dynamic data.

3.3 Sharing the workflows

Finally, sharing an analysis workflow begins a stage of development in which peer feedback can play an important role in improving and extending the research. To prepare a workflow for sharing, metadata are assigned to every component to describe the relevant details for understanding the analysis, such as input values, processor functions, and configuration options for running alternative analyses. Any processes that access repository databases directly or leverage web services created to support the analysis are carefully engineered so that the shared workflow does not pose security problems to the server that provides the web services. Once these issues have been addressed, the workflow is shared on MyExperiment.org, from which we can link papers or data sets that are related to the analysis, allowing us to better close the loop between data and results.

Sharing analysis workflows with researchers outside of our own research group also

achieves the desired transparency of operational specifics that the published papers themselves do not possess, so the workflows also allow more rigorous evaluation of the research. This is especially valuable in an area where many researchers are using the same raw data sources but handling them differently, leading to questions surrounding validity. If researchers routinely provided such transparent details of their analyses, the entire research community could more usefully compare different measures of concepts, indicating another potential long-term benefit from adoption of these methods and tools.

3.4 Lessons Learned

The effort of replicating research with eResearch workflows provided many learning opportunities. In this section, we discuss our observations about the use of eScience for FLOSS research that may have implications for the application of eScience ideas to social science more generally. We found that data is one particularly key issue; although data are readily available in repositories amenable to use with workflows, handling the data is a more significant challenge than we foresaw. In addition, we found that designing for flexibility and transparency eased the collaborative development process, yielding workflow products that are well suited to reuse and repurposing.

3.4.1 Data

In selecting research to replicate, we chose studies requiring data from several FLOSS repositories, and one study that required data from multiple sources. Our success in meshing data from two repositories for the English & Schweick (2007) classification workflow proves that it is possible to leverage the federated wealth of FLOSS data resources, but the experience also suggests that our repository managers will need to consider how to better accommodate this

type of data usage scenario if we hope to support use by a broader population of researchers. We were fortunate to have individuals with excellent understanding of the data repository contents and the skills to employ a variety of methods to retrieve data from different sources, but this is not the case in every research group.

Combining data drawn from multiple repositories was challenging, but we were fortunate that in our research population—FLOSS projects—each project adopts a unique identifier by forge, and these identifiers are not only present but consistent across FLOSS data repositories. Without this key identifier, cross-repository analysis would have been significantly impeded. Unfortunately, the existence of a unique identifier across data sources is likely an exception rather than the norm. Semantic technologies seem particularly promising for improving data retrieval by increasing interoperability across multiple data repositories without requiring structural changes to the repository databases. The challenges we encountered with data indicates this a primary area for improvement; creating and extending a library of reusable components for handling FLOSS repository data will make a significant impact by improving data accessibility if eScience approaches are more widely adopted in the FLOSS research community.

3.4.2 Design for flexibility and transparency

We also developed some useful experience in approaches to workflow development. To maximize the potential uses of the workflows, our goal was to produce analysis workflows that are modular, transparent, reusable, and have low interdependence. To accomplish this goal, we applied two guiding design principles. First, we parameterized all thresholds and variables. This parameterization enables sensitivity testing of the values, where the original research typically used only one carefully selected value. The resulting design of the workflow incorporates more

inputs while permitting maximum flexibility.

Second, we created simple components with low interdependence, following the design philosophy of creating "small things loosely connected." We attempted to reduce each component to just one operation or data manipulation; these small components have maximum reusability and transparency, both in their internal code (for Beanshell or RShell) and in the structure of interdependence of operations. This design approach has numerous benefits, yielding workflows that were more transparent, less system-dependent, and easier to troubleshoot and share.

Modularity yielded several notable benefits. First, by constructing our workflows with many small components, we found it easier to co-develop the components and integrate independent efforts. It also permitted us to easily and quickly change strategies by inserting new components that required minimal adjustment of the existing workflow structure. For example, two researchers independently designed, developed and tested the subworkflows defined in the abstract workflow for the English & Schweick (2007) replication, and then merged our subworkflows and components to create a fully functional workflow (Figure 7). The task of merging the workflows was remarkably fast and straightforward despite the overall complexity of the analysis, and required less than 15 minutes of effort. The resulting workflow, shown in Figure 7, appears quite complex, but the modular structure means that much of that complexity can be hidden.

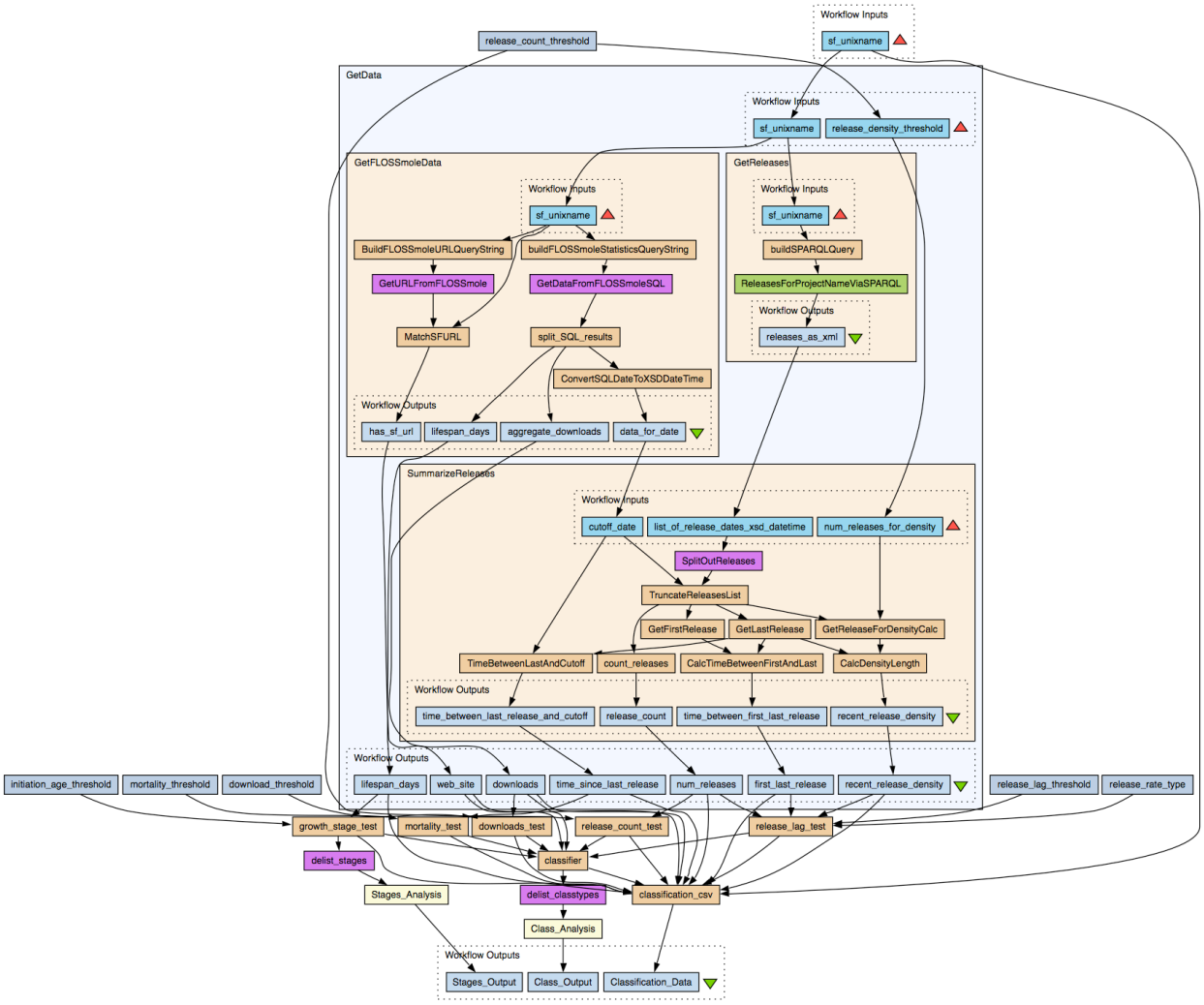

Figure 7: The most recent version of the classification workflow. Note that most of the workflow complexity lies in the "GetData" subworkflow that accesses repositories and prepares the raw data for analysis.

However, modularity must be moderated with good sense; for example, in our classification workflow, each pre-classification criterion check (e.g., are there enough downloads to consider this software useful?) is an independent component but the actual classification determination resides in only one component, which has a much higher internal complexity, but also serves as a human-readable logically complete truth table. In this case, the transparency of the process is improved by containing the whole of the classification in one

component.

Second, the workflow analysis software directly supports a modular development approach by allowing use of a variety of component types in implementation. For example, we created Web services for one of our replications, using SOAP to wrap the existing code from the original research analysis for the replication. The same processes could be achieved by developing local Beanshell scripts to perform the data manipulations, but in this case, it was more expedient to set up the Web service. In addition, using SOAP to repurpose bespoke scripts allows replication in the strictest sense, in which the original data handling is recreated by using the same scripts invoked through Taverna.

Third, modular workflows are easier to debug and execute. At each development iteration of testing and revision, exception cases were identified using Taverna's native process reporting. Inspecting intermediate values passing between many small components allows us to more quickly identify and solve problems, some of which might otherwise cause errors that would be hidden if they occurred within a compound process. The workflow was adjusted accordingly, and when it successfully processed a data sample, it was tested on a larger sample, working up to full sample size or the limits of the computational resources, whichever came first. Besides allowing us to more efficiently identify problems with the workflow design, running the workflow processes on small data samples was important for two additional reasons: we could manually verify the outcomes and therefore were able to reduce our debugging time by testing as many edge cases as we could mock up, and we were also able to get a sense of the computational resources that each workflow would require for performing large-scale analysis.

Finally, the design of the workflow is where variations in operationalization of concepts

can be implemented, transparency and modularity can be preserved for future reuse and auditing, repository data sources can be used instead of re-collecting raw data, and analysis design can be extended. The benefits of these design outcomes are hard to properly value: repository data becomes easier to retrieve and use, providing access to a wider research audience; data handling processes can standardize to best practices based on combined expertise, supporting greater confidence in research findings, while still allowing alternate approaches; and analyses are easily shared, vetted, reused, and extended. However, the ability to modularize an analysis in this way depends on the state of the field, as we will discuss below.

4. Discussion: Issues in adoption of eScience

In this paper, we have described the nature of eScience as applied to the social science study of FLOSS development, as driven by the nature of FLOSS development itself. The research area seems ripe for the application of eScience tools, demonstrated in our case study, which will benefit the field by providing a shared base of data and tools and in the long term, more accumulation of results as researchers learn to build on each others' findings. Indeed much progress has already been made, especially through the FLOSS repositories. In this section, we attempt to generalize from our case experiences to discuss possible implications for eScience in the social sciences more generally.

4.1 Where does eScience work?

In order to understand the broader implications of eScience in the social sciences, it is important to delineate the kinds of research for which eScience approaches are likely to be feasible and useful. For example, we have identified the availablity of common data sources in FLOSS as a driver for the application of eScience practices. On the other hand, much research in the social

sciences relies on survey and interview methods, where there are clear privacy issues in the collation and publication of raw results. Nonetheless, the established practice of sharing survey instruments and content analysis schemes could be formalized into a searchable repository and techniques such as factor analysis could be published as workflows, even if 'dummy' data is used to demonstrate the techniques.

Another useful distinction is between “big science” and “little science” (de Solla Price 1963). “Big science” refers to scientific projects that draw on multiple disciplines to address a broad set of goals, which are often set by a committee that then selects the researchers to carry out the work. Big science increasingly demands eScience methods: more researchers are dependent on the outputs of scientific infrastructure and indeed, the most prominent examples of cyber-infrastructure are found in these fields. The fundamental nature of the data produced in big science has also changed; petabytes of data and teraflops of processing are now normal operating conditions for many natural sciences. Growing arrays of sensors and instrumentation produce more data streams to fuse for analysis, creating additional challenges for researchers. In addition, funders increasingly require that data generated by large research efforts to be shared with the research community more widely, making eScience contributions by “outsiders” an increasingly likely source of new knowledge discovery. Because big science has been working on adapting to eScience technologies for some time, many fields in the natural sciences have developed standards to enable efficient use of the data and technology. While learning these standards is a barrier to entry for newcomers into the field, the standards support a more complete, carefully vetted documentation of the state of scientific practice than most individual research experiences can encompass.

By contrast, “little science” refers to a single investigator working on projects of their

own choosing with relatively modest support, such as a graduate student or two. Such approaches are more typical in many social science fields, though there are certainly exceptions. In little science, the advantages of eScience methods are less clear. Little science is less likely to have standardized work in a way that allows smooth reuse of data and analysis tools. The benefits of modular workflow development are less useful if there is only one person working on the analysis. While in principle the technologies and practices have the same potential for enabling discovery in little science, the reality is that in small research communities, data and technologies have generally been created for the individual research groups' goals, and the infrastructure that enables transferability and interoperability of data and analysis technologies may not be available. It seems likely that little science will require more time to adopt the social infrastructure of practice that big science has already developed of necessity, more coaxing to achieve community buy-in of the principles and practices in the absence of funding-driven collaboration of eScience, and more flexibility built into the standards as they evolve. At the same time, little science has the advantage of the examples of infrastructure developed by big science. Strategic efforts by leaders in specialized fields to adopt and adapt the existing practices and technologies has the potential to produce dramatic results.

A related distinction is between normal and pre-paradigmatic science (Kuhn 1962). Many eScience applications rely on shared agreements about data, concepts, theories and analysis approaches. Such agreement characterizes normal science, but is largely absent in pre-paradigmatic areas. In the absence of such agreement, there will be little interest in reuse of data or workflow modules for particular analyses. Many social sciences are quite paradigmatic (e.g., economics) but others are less so. It may be that such agreement is achieved only within certain subfields, as we suggest may be the case for FLOSS research.

4.2 Social issues in the adoption of eScience in FLOSS

In the discussion above, we have mostly focused on the technological aspects of eScience. However, technical tools are only half of an infrastructure. To make the technology successful will require addressing a set of social issues that encourage or discourage the use of the infrastructure. When applied to data or knowledge, these issues raise a broader set of questions, such as the issues around policies for data curation to ensure that data have the necessary documentation and are of acceptable quality to be reusable. In many cases, a considerable amount of tacit or domain knowledge is needed to make sense of the data, posing an obstacle to broader use. Long-term funding must be obtained to preserve and migrate data and workflows, and to make them accessible to researchers. A common objection to data sharing is concerns about the privacy of research subjects. The use of public data sources avoids some of these concerns, though FLOSS data poses interesting ethical questions about appropriate privacy policies for aggregated data that is already available elsewhere on the Internet. A related issue is the intellectual property concerns about storing and redistributing such data.

Beyond these challenges, the most important challenge will be motivating individual researchers to participate, both in using and contributing raw data, intermediate results and analyses. Developers of the current repositories described above have made data available, driven in part by the prevailing ethos of openness in the communities they are studying, but for eScience approaches to be more broadly adopted, further incentives seem necessary. To understand these issues requires an institutional level of analysis, taking into account how current work practices are situated in a variety of settings and organizational structures. Given this context, motivations for participation might include policies about rewards for sharing, e.g., citations, letters of recommendation or generalized reciprocity as well as more coercive

enforcement via reviewing or funding policies.

In addition, research practices that extend beyond one's own desktop or research group requires shared goals and direction for a larger research agenda which may be lacking where the research community has not defined a set of accepted grand challenges. Without a common direction for future research, the lack of funding imperative for collaboration in small sciences means that there is little incentive for individual researchers and small research groups to reach out to their peers. Ironically, while big science funding has required collaboration due to the high costs of instrumentation for data collection, reduced funding to little science research may serve a similar function by promoting data reuse and broader collaborations to make the most of increasingly limited fiscal resources.

A key question in developing collaborations asks who can participate and what differential benefits they may gain. On the one hand, eScience presents the opportunity for increased participation in science, since researchers can access data, knowledge and collaborators from wherever they are, rather than having to be where those resources are located. On the other hand, since there is a cost to developing, maintaining and using cyber-infrastructure, it may be that only well connected and well endowed institutions can fully participate. Greater openness through eScience methods may confer advantage unevenly, but shows good potential for more meritocratic criteria for evaluation of research contributions. In a more ideal world, the development of robust citation mechanisms to link data, analyses, and publications may lead to a changed perception of the kinds of work that comprise research.

5 Conclusion

Through our workflow development efforts, we have demonstrated the potential capacity for

extending the analysis and scale of the research by implementing analysis design to extend prior research efforts and enabling analysis on a potentially larger scale than the original research. The workflows and analysis outputs are proof of applicability of methods for the field, and while reproducing research as published in papers can pose significant challenges, we find the results of these initial forays into developing workflows for FLOSS research encouraging.

Acknowledgments

This research was partially supported by US NSF Grants 05-27457 and 07-08437. Thanks to Megan Conklin for permission to use the figures from her 2004 presentation for comparison.

References

Antoniades, I., Samoladas, I., Sowe, Sulayman K., Robles, G., Koch, S., Fraczek, K., and Hadzisalihovic, (2007) A. "Study of Available Tools " D1.1, FLOSSmetrics.

Cataldo, M., Wagstrom, P.A., Herbsleb, J.D., and Carley, K.M. (2006) Identification of coordination requirements: Implications for the design of collaboration and awareness tools. *Conference on Computer Support Collaborative Work.* Banff, Alberta, Canada.

Conklin, M. (2004) Do the Rich Get Richer? The Impact of Power Laws on Open Source Development Projects. *Proceedings of Open Source*.

Conklin, M., Howison, J. & Crowston, K. (2005) Collaboration Using OSSmole: A repository of FLOSS data and analyses. *Symposium on Mining Software Repositories.* St. Louis.

Conley, C. A. (2008) Design for quality: The case of Open Source Software Development. New York, NY, New York University.

Conway, M. E. (1968) How do committees invent. *Datamation,* 14**,** 28–31.

Crowston, K. & Howison, J. (2005) The social structure of Free and Open Source Software development. *First Monday,* 10.

Crowston, K. & Scozzi, B. (2008) Bug fixing practices within free/libre open source software development teams. *Journal of Database Management,* 19**,** 1-30.

de Solla Price, D.J. *Little Science, Big Science* Columbia University, New York, 1963.

English, R. & Schweick, C. M. (2007) Identifying Success and Tragedy of FLOSS Commons: A Preliminary Classification of Sourceforge. net Projects. *Emerging Trends in FLOSS Research and Development, 2007. FLOSS'07. First International Workshop on***,** 11-11.

Herbsleb, J. D. & Grinter, R. E. (1999) Splitting the organization and integrating the code: Conway's law revisited. *the International Conference on Software Engineering (ICSE '99)*. Los Angeles, CA, ACM.

Howison, J., Conklin, M. & Crowston, K. (2006a) FLOSSmole: A Collaborative Repository for FLOSS Research Data and Analyses. *International Journal of Information Technology and Web Engineering,* 1**,** 17-26.

Howison, J., Conklin, M. S. & Crowston, K. (2005) OSSmole: A collaborative repository for FLOSS research

data and analyses. *1st International Conference on Open Source Software.* Genova, Italy.

Howison, J., Inoue, K. & Crowston, K. (2006b) Social dynamics of free and open source team communications. *Proceedings of the IFIP 2nd International Conference on Open Source Software, Lake Como, Italy*.

Howison, J., Wiggins, A. & Crowston, K. (2008) eResearch workflows for studying free and open source software development. *Fourth International Conference on Open Source Software (IFIP 2.13).* Milan, Italy, Springer.

Kemerer, C. F. & Slaughter, S. (1999) An Empirical Approach to Studying Software Evolution. *IEEE Transactions On Software Engineering,* 25.

Kuhn, T.S. *The Structure of Scientific Revolutions* University of Chicago Press, Chicago, 1962.

Nentwich, M. (2003) *Cyberscience: Research in the Age of the Internet,* Vienna, Austrian Academy of Sciences.

Roblex, G., Amor, J. J., Gonzalez-Barahona, J. M. & Herraiz, I. (2005) Evolution and growth in large libre software projects. *Principles of Software Evolution, Eighth International Workshop on***,** 165-174.

Robles, G., Koch, S. & Gonzalez-Barahona, J. M. (2004) Remote analysis and measurement of libre software systems by means of the CVSanaly tool. *the 2nd ICSE Workshop on Remote Analysis and Measurement of Software Systems (RAMSS), 26th International Conference on Software Engineering.* Edinburgh, Scotland.