Meanings and Boundaries of Scientific Software Sharing

Xing Huang[1], Xianghua Ding[1], Charlotte P. Lee[2], Tun Lu[1], Ning Gu[1]

[1]School of Computer Science
Fudan University
Shanghai 201203, China
{xinghuang, dingx, lutun, ninggu}
@fudan.edu.cn

[2]Human Centered Design & Engineering
University of Washington
423 Sieg Hall, Seattle
WA 98195, USA
cplee@uw.edu

ABSTRACT
In theory, software, like other digital artifacts, can be freely copied and distributed. In practice, however, its effective flow is conditioned on various technical and social factors. In this paper, drawing on ethnographic work primarily with a bio-informatics research team in China, we report on meanings of scientific software sharing as embedded in social practices of learning, apprenticeship, membership, publication, and reputation. We illustrate that while free flow is important, boundary management is equally important for the effective travel of software to its appropriate destinations. Our study highlights a number of issues that are important to consider for effectively supporting sharing and collaboration in science.

Author Keywords
Software sharing; scientific collaboration.

ACM Classification Keywords
H.5.m [Information Interfaces and Presentation]: Miscellaneous; H.5.3 [Information Interfaces and Presentation]: Group and Organization Interfaces - Computer-supported cooperative work.

General Terms
Design; Human Factors.

INTRODUCTION

Sharing is central to contemporary science. In the *open science* system famously characterized by Merton [31], scientists openly publish their scientific findings and knowledge, and are rewarded with academic credit. This is in great contrast to previous eras when knowledge of nature was considered to be sacred secrets (e.g. ancient alchemy) [15]. Although contemporary research groups still self-consciously conceal research results to compete for recognition of discovery priority [19, 28], sharing is generally positively perceived and actively encouraged [16, 50].

CSCW '13, February 23–27, 2013, San Antonio, Texas, USA.

In CSCW in particular, fostered by the development of IT, new research programs exploring the use of IT to provide new sites for scientific sharing and collaboration are on a rapid rise, under terms of "collaboratories", "cyberinfrastructure", "eScience", or others [20, 27, 34]. With its great potential to enable communication and the move of information rapidly and relatively inexpensively across time and space, there is a high expectation that IT can substantially speed up and widen the access of resources among scientists [27]. Three interrelated aspects of sharing are typically explored: instrument sharing focusing on the sharing and access of key or expensive physical instruments remotely [26, 27, 32], data sharing concerning making scientific data openly available for verification and other investigations [6, 8, 48], and knowledge sharing focusing more on the exchange of talents, expertise and effort [10]. Studies of these projects show that, for scientific sharing, while IT provides great potential to overcome distance barriers, there are still many other socio-cultural barriers, such as cross-institutional barriers [10], disciplinary and institutional cultures [11, 44], credit systems [7], and so on. These studies point to the importance of understanding relevant social and cultural practices to effectively support scientific sharing with IT.

While in these programs software as part of the *infrastructure for sharing* has been extensively explored, software as *an object for sharing* itself is largely overlooked. However, understanding software sharing is of great importance for scientific collaboration and innovation, mainly for two reasons. First, software plays an increasingly critical role in scientific research, including but not limited to data storage, integration, analysis, processing, representation, and others. Not only does it make work easier, faster or more efficient, but also it changes the very nature of doing research, such as in emerging new interdisciplinary fields including bio-informatics. Second, software is distinct from other aspects of sharing in fundamental ways, and warrants separate study. As with other digital artifacts and unlike the more physical instruments, "*software can be copied and distributed at essentially no cost*", which provides unlimited potential for sharing [22]. At the same time, software is defined as a set of programs directing what to do and how to execute certain tasks on a computer, which is essentially different from data or knowledge. So it may play very different social roles in

science, which can greatly influence how it is shared and how we can support it.

In this paper, we report on an ethnographic study at a bio-informatics research center in China, with an aim to deepen the understanding of software sharing practices in science, and inform the design of computing systems to better support them. We define scientific software broadly as *all kinds of software that are used to support scientific research*. With *scientific software sharing*, then, we concern ourselves with social practices related to the creation, navigation, use, selection, and distribution of all these sorts of software, and seek ways to better support them. By focusing on this new emerging discipline, bio-informatics, in which software plays a central not a secondary role, as well as China as the particular context, we hope to be able to highlight some of the issues that might be obscured in more traditional or familiar sites, shed light on scientific sharing, and derive design implications.

RELATED WORK

For software sharing in general, probably the most studied topic is open source software. Extensive research attention is paid to the contribution and motivation mechanisms behind the development of open source software, including how the social context and social networks influence the project licensor's decision [45], how the choice of license influences people's passion for contribution [12, 35, 38, 39], and how meeting one's own needs motivates people's participation [9]. Studies also show that factors such as the readability of code and manual documents are important for user acceptance [18]. Others describe work practices, development processes, and community networking of open source software communities [13, 40]. While open source software plays a large part in scientific research, however, how it is used or shared in science still has not been closely examined.

For scientific software in particular, sharing has also been of concern for a while, especially in software research. Many found that scientific software is hard to maintain and is limited for long-term use [1, 23, 36]. Some point out that scientific software documentation is problematic, including comments in the code, user guides, manuals, and so on [33, 37, 42]. Others attribute scientific software problems to developers' lack of software engineering training [24]. This line of work suggests that although software can be copied and distributed at essentially no cost, software quality issues can prevent software from being widely applied.

More recent research focuses on developing middleware, adapters, and protocols to support "collaboration transparency", through which a single-user application can be shared between more than two people [5, 47]. By combining these sharing approaches with the development of grid computing [17], web services [4] and cloud computing [3], software can then be delivered by means of services, in hopes of further fostering software sharing across time and space.

Overall, scientific software sharing has mainly been examined from a technical perspective and few have focused on its social practices. The recent exception is Howison and Herbsleb's work on how incentives impact software production [21, 22]. Similarly from a social perspective, however, our work is distinct by looking into details and meanings of software sharing with respect to broader collective social practices, beyond software production and the issues of incentives.

RESEARCH SITE, BACKGROUND AND METHOD

Our study is primarily based on a center for bio-informatics technology in China. Bio-informatics is defined as the application of computational techniques to understand and organize the information associated with biological macromolecules. Bio-informatics emerged in recent years as the surge in biological data demanded computation to handle it [30]. The center was established in 2002, and as a non-profit organization, the mission of the center is to promote the advancement of Life Sciences, Biotechnology and the Biopharmaceutical industry by engaging in life science database construction and bioinformatics software development. The center is organized with a number of research teams focusing on various topics including functional genomics, system biology, translational medicine, pathogen genomics and so on.

As with most research teams in China, the research center is organized and managed in the form of Tidui (echelon, "梯队"). A Tidui is usually composed of one senior faculty, several junior faculty, and many graduate students. As the team is often large (20 or more), its management also tends to be hierarchical, in that new junior students are directly mentored by senior students as their ShiXiong/ShiJie (师兄/师姐，senior brother/sister apprentice) for everyday help, all students are managed by junior faculty as their team leaders, and all students/junior faculty are advised and directed by the senior faculty. As a research center with no graduate program, our site, though, is different from other research teams in universities, in that there are no graduate students officially affiliated with the center. However, as the head of the center is also a professor affiliated with a number of institutions and universities, his students from these institutions and universities are also involved in various projects at the center. So the center is a spatially distributed group as most junior students still need to spend time on their own campuses for coursework, and can only be at the center occasionally.

For this study, we have mainly engaged with the team led by Liu, a PI in the center. The team's research area is translational medicine, involving proteomics related bioinformatics, transcriptional regulation and oncology related bioinformatics. Their main focus and contribution is on developing software for analysis, and is mainly based on

open data (which is important for the independence of the discipline), except for collaborative projects with other teams (e.g. a team from a hospital) who provide data and they apply algorithms to analyze the data. We have been collaborating with the team to develop an open platform to support collaborative protein analysis by integrating related software and communication tools, which partially motivated our investigation on software sharing in science in the first place.

In building the platform over the past two years, we attended monthly meetings with the team, were often invited to attend their seminars, occasionally engaged in discussion meetings with other related teams, and often worked together with the team to fix technical problems. These were the main forms of our participant observation at the center, and, in total, we spent 150 hours in the field and collected 23 meeting memos. Through this process, we noted many factors that have shaped how scientific software circulates within and without the team.

Partially motivated by our observations, we conducted more in-depth interviews to directly inquire how software is shared in practice. We completed semi-structured interviews with 15 informants in total. Interview times ranged from 25 to 120 minutes, with 50 minutes the average. In our interviews, we asked about the participants recent research projects, and details of how scientific software was produced, selected, used, and distributed with regard to these projects. When possible, concrete cases and examples were sought. Of the 15 informants, 13 were from Liu's team, and 2 were from their collaborator – the Tidui led by Ceng from another research institute. Ceng's team focuses on wet lab experiments, original data production, and some preliminary data analysis. Most of our informants only write scripts such as R, Python and Perl to integrate software for analysis, and only one informant (from Ceng's team) could be categorized as a *professional* end user developer who writes programs in Java or C++, and his main responsibility was to develop software for the rest of the team to do research. Our informants' positions were: 4 master students, 2 junior Ph.D. students, 4 senior Ph.D. students, 1 graduated Ph.D. student (now a post-doc abroad), 3 regular full time researchers, and 1 PI (Liu). All interviews were conducted in Mandarin Chinese, and the participants and interviewers were all native Chinese speakers. Quotes used in this paper were later translated to English.

We then applied a grounded theoretical approach for the analysis of the data [46]. We started analysis while the data was being collected. Through open and axial coding of the data, themes started to emerge (while our participatory observation greatly influenced our interview questions and analysis, coding and analysis was primarily based on our interview data). In this paper, we will use representative individual quotes to illustrate our points. All of the names used are pseudonyms.

TYPES OF SOFTWARE IN BIO-INFORMATICS

In bio-informatics, the development of software for the management and processing of biological data forms the main contribution of their research work. So unlike other disciplines [22], software production plays a central, rather than a secondary, role in their research. Specifically, from our study, we identify four types of software involved in their daily research work: *published academic software*, *open source software, homemade scripts, and commercial software products*. *Published academic software* refers to software that is published in the form of a paper that contains relevant rationales, methods, algorithms, and information about the software. Paper publication indicates novelty and significance of the software to the community, and corresponding software is usually made publicly available once published. *Open source software,* freely available in the form of source code, can be used, distributed and modified by anyone under an open-source license, and is found to be heavily used in their research. *Homemade scripts*, written to do particular data processing, or to stream different steps of software work together, are also commonly used but usually only written for personal purposes, not intended for sharing. *Commercial software products* are relatively the least used, since they are too expensive for individuals to afford, and many times, there are free alternatives available in terms of *open source software* or *published academic software* (often as good and are well recognized by the community). Of course, these categories are not entirely exclusive, and occasionally, one type can be transitioned into another. For example, some homemade software, originally written for personal or internal use, may later be published or contributed to an open source software community. Overall, we found while researchers heavily rely on open source and published academic software, always write and use scripts in their research, and occasionally use commercial software products, their main contribution or output is the published academic software. Next, we will elaborate in more details of how and why software is circulated, and how a range of boundaries are leveraged or contrived to regulate its flow.

MEANINGS OF SCIENTIFIC SOFTWARE SHARING

In our study, we believe two social entities are central to characterize the meanings and mechanisms of scientific software sharing: the local team, and the research community. The local team (or Tidui) in China is the "internal us", in which resources are centrally managed and coordinated. It is largely a people-oriented unit, where members may not necessarily share the same space or the same institution, but they all share the same leadership and management. In our case study, for example, most of our student informants were from different universities and institutions, but were all advised by the same professor and managed by the same team leader. Moreover, it is also a basic unit where junior members learn and grow into qualified and independent researchers, through "peripheral participation" or apprenticeship [29]. The research

community, on the other hand, is more a topic-oriented social configuration, where members are connected by common interests, mutual engagement and a shared repertoire [49]. In this section, we will explicate why and how scientific software is shared with respect to these two social entities: internally with the local team and externally with the research community.

Apprentice and Membership: Sharing Internally with the Local Team

Within the local team, characterized as a close-knit and highly inter-dependent social entity where members mutually help each other and are generally aware of what each other is up to through regular group meetings and project participation, software and related information are generally well shared. Especially, homemade scripts and work-in-progress academic software are mainly shared within this unit and rarely travel beyond it. Particularly, the meanings and mechanisms of software sharing are highly related to apprenticeship and membership of this group.

Experience Sharing through Apprenticeship

As mentioned, one important component of a research team is learning/training, mainly through apprenticeship. In our case study, we found that it was common for new junior members to be assigned with more senior members (usually above the third year in their graduate program), or Shixiong/Shijie in their words, for everyday help, advice, and guidance. New junior members, by engaging in marginal work and with the help of Shixiong/Shijie and advisors, learn the tricks of the trade of doing research, and obtain relevant research skills including program writing. So learning, training and passing down experiences through apprenticeships are inherent themes that define the identity and culture of this group, which in turn, have greatly shaped how ideas, information, resources and software circulate within it.

First of all, by sharing their experiences, Shixiong/Shijies work as important gatekeepers to help them navigate through and choose from all the publicly available software out there (e.g. open source software and published academic software). In the field of bio-informatics, there is a consensus that published or academic software should be made publicly available for free. Throwing software out there, however, does not directly lead to effective use of it. As a matter of fact, with all of the *published academic software* as well as other *open source software* made available, it becomes a big challenge in terms of how to choose and navigate through it, especially for new members. Almost all informants reported that Shixiong/Shijies were of critical help in this regard. Li described a typical case, "*oh, Shijie said… the Shijie who helps me… she said that was a common method. So I tried it.*" So by simply following what Shixiong/Shijies use or what they recommend, new members learn what is commonly used and practiced in the community.

Further, Shixiong/Shijie's experiences are also important for them to build up programming skills. They help by engaging in discussions, checking software codes, and providing relevant resources. Hu commented that discussion with his Shijie was helpful, "*mainly in terms of thoughts. She will tell you how to think about it, but for details, in terms of how to write, you should do it yourself.*" Wu reported that his Shixiong's hints such as a package's name were useful for him to locate relevant resources. Others provided relevant documents and examples for new members to learn, as Wu said, "*he directly gave it to me… all the documents corresponding to each package, and there were examples and others, and I studied them myself.*" Of course, sometimes, it involved direct software sharing, "*I checked with my Shixiong, what functions or what packages shall be used for drawing it this way, and he said, 'I would give you this script and it could solve the problem', so I just took it and used it.*" As we can see here, how software is shared in this context is clearly associated with experience sharing between apprenticeship relationships. And, with more and more open source and academic software becoming publicly available, experience sharing for software navigation might be more important than the sharing and distribution of copies of software themselves.

Interestingly, the motivation to learn also shapes their decisions regarding whether to reuse others' software or to rewrite it on their own, especially for simple scripts. Script writing is an integral component of learning and training for junior members to grow into qualified bio-informatics researchers. Consequently, sometimes, even if there were scripts available to meet their requirements, the junior members would still opt to write one themselves, as Lian described, "*The problem was too simple to use others' scripts, and it was also a way for me to practice my script writing skills, so I wrote it all by myself.*" It reminds us that in a culture where learning/training is an inherent component, directly sharing a copy of software itself might not always be desirable, but rather, sharing relevant resources for learning and software making might be more helpful.

Moreover, we note that sharing through apprenticeship is mainly through the "request-and-give" social protocol. That is, junior members request and their Shixiong/Shijies respond with relevant information, as part of their mentoring responsibilities. It is also important to note that such a "request-and-give" social protocol is critical for experience and software sharing in this context. Considering an important component of knowing is embedded in what we do, tacit, subconsciously held [41], and not readily available for transferring to others, the request-and-give protocol, then, provides an important mechanism to turn the tacit knowing into explicit and transferrable information.

Software Sharing as Part of Membership

While sharing through apprenticeship can be accounted as part of responsibility, sharing among peers works more as markers of membership. Dourish and Anderson, drawing on social studies of secrecy, note, "*the practices of keeping and sharing secrets are ways in which affiliation and membership are managed and demonstrated*" [14]. By analogy, in our study, we similarly found that the practices of keeping and sharing software are ways in which affiliation and membership are managed and demonstrated.

Sharing as part of membership is most clearly seen from the sharing of *homemade scripts* within the team. Bai, when asked whether he would give a piece of software to his teammate, answered that being part of the same lab justified this sharing, "*I will give for sure. We are all from the same lab after all. If someone emails me, I will give it to him for sure.*" Interestingly, he further stressed "*There was nothing so secret, and you don't have any copyright*". It suggests to us that, for simple homemade scripts, since they are not as valuable (in terms of innovation and labor taken) as some academic or commercial software, there is little obstacle to constrain their flow. Yet, at the same time, it is exactly the low value that confines its capability to travel too far. As a result, sharing of homemade software is mainly contained within the local team.

In our case study, we did not find any central place to store and manage the homemade software, and sharing was also mainly through the request-and-give social protocol within the team. Yin mentioned a case, "*later, another teammate also came to me and asked for it. As he was doing similar things, so he asked and I gave it to him."* When asked whether software sharing was always very open in her lab, she responded by explicitly pointing out that homemade software was mainly shared through private requests, email or verbal, *"not like that, but if someone asks in private, I will give. It is not like we have a public platform for it, not like that.*" Our data further suggests that many factors may contribute to the non-use of an open and centralized place for storing and sharing for this kind of software. First, most of the software is written for personal use to address a problem at hand, and it is unexpected that one day others may also find it useful. Second, as it is mainly written for personal purposes, it may not be so readable or understandable for others to use the software [24]. So sense of belonging is important for the sharing of homemade software within a team, which is similar to open source software [2]. However, unlike the production of open source software, which is communal with centralized management and coordination [38], the production of scientific software is mainly base on distributed individual effort.

For the request-and-give social protocol to work among peers, awareness of what each other is doing is a key component. As expected, regular group meeting is an important means for them to learn what each other is doing. Zan described, "*We will present (our work) in our group meetings. There are so many people in the lab, and generally we know what each other is doing.*" Qiangrui revealed a typical situation for software sharing in his group, "*you will talk about your work and your progress in your group meetings. And there will be people who may encounter the same problem for sure. If you have solved the problem, they will ask you how you solved it. And you will say, oh, I wrote something. Then he would say 'could you share the code with me', and I would give the code*." It suggests that, for the effective circulation of homemade software, using computing to enhance awareness of what each other is doing can be valuable here, especially when members are quite distributed, as in our case study.

Publications and Natural Software Selection: Sharing Externally with the Research Community

For software with certain innovation and scientific significance, it may appeal to wider audience, go beyond the local environment, and become academic software open to the external research community through publication. In this section, we are mainly concerned with the circulation of *published academic software* within the community. Unlike the local team, the research community is largely a practice-oriented social configuration. While members may share common research interests and engage in similar research activities, there is no official social affiliation involved, and "*practice is the source of coherence of a community*" [49]. As an academic community, it is also dominated by a "reputation economy based on substantial publications" [22]. Howison and Herbsleb, in their study of the production of software, point out that "*Software is a secondary player in the world of scientific work*" [21]. Fortunately, in bio-informatics, scientific software plays a central, not a secondary, role, and substantial part of scientific software is produced and shared exactly through the reputation economy and the publication system. Authors gain academic credit directly from publishing the software, and software publications are well recognized as their primary contributions. Once the software is published, papers, citation mechanisms, reputation systems, as well as the culture and norms of the research community start to take their way to shape the evolution and circulation of software. As such, it is impossible to talk about the circulation of academic software without talking about the mechanisms and functions of publications.

Sharing through Publications

The circulation of academic software aligns well with the circulation of publication. In the bio-informatics community, of our study especially, there is a consensus that once the paper is published, relevant software should be made publicly available in one way or the other. Now it is not the issues of availability, but the issues of visibility, documentation, awareness, navigation, evaluation, selection, reputation and others that have become critical in shaping

the circulation of academic software, and all of these are dependent on the publication mechanism.

Firstly, the paper reviewing and publication process helps to normalize software and relevant documentation, and sets a bar for the selection of software quality for the community in the first place. All academic software is associated with at least one paper, which contains relevant rationales, methods, algorithms, and more, as part of the norms for publication. Bai put it, "*the essence of documentation about a piece of software is in its published paper*". In a sense, publications serve as a persistent shared repertoire [49] of knowledge, information, references, authorship, and others, and they can reach out to a wider audience far beyond the space and time constraints. While the lack of documentation is commonly complained as one of the main factors constraining the widespread use of scientific software [37, 42], the correlation between software contribution and paper publication together with the reviewing process seem to provide a way to relieve this issue.

Further, visibility and awareness obtained through publication is important for the wide distribution of software. Our informants were fully aware of how important publication was for their software circulation. Huang described, "*it (widespread use) is mainly dependent on paper publication. If it cannot be published, others will not use it.*" It is especially helpful if it is published in high quality journals, as Liu mentioned, "*I never make specific efforts to recommend it, except I attend conferences, and we have lectures. But if we have good papers, that is if we publish our software tools in the internationally well known journals, it is a good recommendation in itself.*" On the other hand, almost all of our informants reported that they were highly reliant on publications to learn about methods, software, and relevant knowledge. When talking about software selection, Ji said, "*Usually, when the paper is published, they will compare their own tools with other previous tools. Usually, that will be part of the paper. So you can take a look, and choose one that might be used more often.*" Some will particularly go for overview papers for this purpose, as revealed by Hu, "*if you want to do phosphorylation, there will be overview papers about phosphorylation for sure, and usually they will mention (relevant software tools). You will know what are out there after you read a few*." As people turn to the publication mechanism for software awareness and selection, publication and software circulation are clearly correlated.

Moreover, publications can also serve as communication channels through which demands and supplies are connected, and it may further drive the evolution of software. Bao's is a case in point. He published a paper on a piece of software, and then received several requests for more features, which motivated him to upgrade the software in the end. He said, "*others read your paper, and emailed to us, asking whether we can do some predictions for them, something they are working on. I got a few of these emails, so I developed (a piece of software) for all. It is publicly open to everyone.*" This piece of work also resulted in a publication. So publications are not merely a shared repertoire, but also communication channels through which people of similar concerns are connected.

Natural Software Selection

With more and more academic software made available, how to select high quality software becomes an issue. After all, not all published software is of equal quality. Actually, we often heard our informants complaining how some people only care about publishing the paper and never care about maintaining the software once it is published, which greatly limits the long term applicability of the software. Further, as mentioned, although in the bio-informatics community, most published software is theoretically publicly available, however, whether it is practically applicable or not is conditioned by many technical factors. For example, the extensibility of software could be an issue. Since it was mainly developed to address a specific question, and only tested with limited data, it may not be applicable or extensible to other situations [33, 43]. Besides, user interactivity also plays a role. Often a GUI is provided to make the software more user friendly, however, we found its demand for user interactivity often meant less usable for more skilled programing users. Bai reported he preferred command interface over GUI, "because the one with *GUI can't run on the server…because, the matter of computation…it is only part of the whole work procedure. If you write a whole work procedure in the script, you don't need to watch for it, otherwise, for example, if there is GUI, it will stop and you need to click for it to continue, which is too much trouble*." Other issues such as compatibility, performance, and the lack of maintenance could also influence its widespread use. So publication itself does not ensure good maintenance, technical compatibility or wide applicability, which are critical for the circulation of software.

In this regard, citations together with paper publications provide a natural software selection process through which high quality and more sharable software can stand out. Qiangrui described how citation worked as a natural selection mechanism, "*if you want to solve a problem, you will check the literature. When you look at the literature, you may see 5 or 6 software tools that can do the same thing. You can try, and you will find which tool is good or not good to use, in terms of usability, interface, analysis capability, and so on. You can feel it. After you feel it, you choose the good one to use, and you will cite corresponding paper of that software in your paper. The tool… the more people use, it will gain more reputation naturally. For others that are not so good, it will be eliminated naturally.*" A high citation count is perceived to be highly correlated with good quality software, as Liu put it, "*usually we use the best software in the field, the most used software, which*

is the most cited in the literature." When asked how he knew which software was the most used, Bao answered, "*for example, like, how many times the (software) paper is cited, things like that.*" In a word, more citations mean it is well tested by more people and is more stable, and our informants reported that they almost always chose those well cited and well recognized software to use, so that the quality of the software can be ensured.

Besides, big names in the community also have great impact on the selection process. Zan described, "*We use what Danius (*magnates*) use or what good papers use. In the field of biology, those very big projects, like the thousand people gene program…what sequence software they use…and we will keep consistence with them. If the paper is written this way, there will be less chance for them to argue, otherwise, it may be troublesome, if you do it your own way.*" It is interesting to note in this case, that the function of reputation is not only to find the high quality software, but also to reduce unnecessary trouble or argument. Lian mentioned that the reputation of other elements, such as corresponding databases or journals or developers, were also important here, "*For this online software, there is a very good database connected to it, quite authoritative. If you do things with it, it will be more authoritative.*" As such, academic software is also part of the reputation economy of academia. Indeed, the circulation of software is very much like the circulation of papers, and only the best will stand out and have wide impact.

BOUNDARIES AND SCIENTIFIC SOFTWARE SHARING

In the last section, we discuss how software and related information are freely shared within the local team and the research community, and how meanings and mechanisms of sharing are embedded in respective social practices. However, software sharing is not always and everywhere free, but rather, many times there are obstacles involved to prevent it from widespread circulation, especially for work-in-progress and commercial software. Issues of boundaries are related to scientific software sharing mainly in two senses. First, software sharing involves tensions between sharing and control, to protect Intellectual Property but also to prevent immature software from wide distribution, and boundary management is a means to resolve it. It is the boundary at which determinations are made about whether to give the software (or the right to use the software) or not and how. Second, there are existent boundaries that regulate the free flow of software, including social boundaries (e.g. between different teams) and technical boundaries (e.g. between different networks). Now we will turn to issues of boundary as an aspect of software sharing, and explicate how tensions arise and how boundaries are managed through contrived means or along existent social or technical ones.

Tension and Boundary Management

Our data shows, tensions between sharing and the simultaneous need to control often arise in the sharing of scientific software, and strategic boundary management is key for its resolution.

Firstly, tension arises when software is still in its production. On one hand, as we know, to make software truly useful, it is important to involve users in the software making processes, the classic participatory design approach [25]. On the other hand, since it is still in production, its wide circulation should be limited. In our case study, we found several successful software tools were directly resulted from involving end users in the production process. For example, one of Huang's published software tools, BuildSummary, was clearly driven by the research needs from his own lab. According to his teammate Qiangrui, this software tool took 10 years to evolve. Over this period, new requirements were continuously fed into its production before it was ready for publication. PFind, a protein identification software tool that was developed by a computing institute in China, was also resulted from its close user involvement from the very beginning. Each year, they sent developers or even had them work at the users' site (including Liu's and Ceng's teams) for one or two months to collect requirements and address issues encountered. After being developed over 10 years, PFind has finally become powerful enough for commercialization right before this writing.

However, while reaching out and involving users are important for software evolution, its distribution should be limited since it is still a work in progress. We found a range of means was employed to balance this tension and manage the boundary. For example, Huang chose different media to share based on whether the software was mature or not, "*If it is not mature enough, or if it is only for mutual (idea) exchange, I will send it over email. And if it is sufficiently mature, they can download it online.*" The limited end-to-end email exchange, rather than the online web, was then chosen as a more appropriate means to share at its early stage. Similarly, for PFind in production, the boundary was strategically managed by socio-technical means. Liu reported a case. She once brought 20 hard copies of PFind to an international conference for academic exchange. The form of hard copy became a way to limit its distribution while at the same time encouraged some to try it, as Liu put it, "*yeah, for example, I took 20 copies with me… That is a limitation. It is not like you can download it for free*." Moreover, the copy recipients were also asked to contact the authors for license if they wanted to try it, "*At that point, it was mainly to encourage people to try it, and we told them, they should contact us and they can only use it after we assign a license code.*" Segal mentioned, scientific software development is iterative [42], and user participation is important over the process. But for scientific research, little mistake may cause great impact on

results, and boundary management is critical here when software is still in production.

Secondly, there is an inherent tension for the circulation of commercial software. For commercial software to be financially successful, broad visibility is important. And yet, to protect its economic interests, its availability shall be well controlled. As a matter of fact, as software can be freely copied and distributed, it makes the management of this boundary trickier. As mentioned, for academic software, publication was an important means to increase visibility. For commercial software, visibility is mainly achieved through active software promotion. Liu observed that software promotion was fairly common at conferences or other venues. For example, she reported, in good international conferences, such as the 3rd Asian Forum on Proteomics, a lot of people would introduce their software in detail so that more people would know the software and use it, even for software that was well known already and where people had already been using it. She also revealed that it was common for salespeople to come to their center and provide free courses and training sessions on their software, in a hope that they would buy it. Other promotion strategies include distributing copies at conferences with limited functions or with several months of probation, or sending copies to famous researchers, etc.

At the same time, for commercial purposes, the availability of software also needs to be carefully controlled. One common strategy reported by our informants is that software is provided for free first as an incentive to try, and then they will charge for upgrading it later. Liu described, "*We've been using it, and after we've used it for a while, all of sudden, they start to charge for fees. If you would continue to use it, you may buy it*." For PFind, the boundary is managed by binding the software to a hardware based license code. Qiangrui said, "*They gave a license to us, and it was a machine code. That is, if you want to use it on your computer, they will send you a copy of software, and it will read your machine's code, which will then be returned to them to form a serial code, something like that. After you get that code, you can start to use it on your computer.*" Interestingly, it is exactly because software can be freely copied that extra effort is contrived to manage its distribution. It sounds like a contradiction, but considering the extra work and cost commercial software takes to offer high-quality services (e.g. performance, stability, usability, etc.), the contrived boundary starts to make sense.

Thirdly, there may be tension concerning the control of unwanted use of software. Software, unlike hardware and others, is more readily open for secondary development, so ownership of a copy often means a range of things they can do on top of it, especially for skilled programming users. Some of these might not be intended or desirable by the original developers. Social agreement is one means to address this tension. Liu reported a case where she got a piece of software from one of her collaborators abroad, and they signed a contract to keep it from unintended use, "*(the contract terms include), for example, it can only be used for academic purposes, and we don't have the right to transfer the copy to others. That is, we can only use it ourselves. Because, it is like a mirror site, if you want to use it yourself, you can use it on his website, but you will have limited functions. That is you can only search, something like that. But if I install the database locally, I can make improvement on it, or, you can even localize it into Chinese.*" It is similar to license use in open source software to define the users' rights to modify and redistribute the software's source code. With the use of social agreement, it is hoped that the original intention, e.g. for academic use only, can go unchanged over software distribution.

Existent Socio-Technical Boundaries

As illustrated, since software can be copied for free, many times, extra effort is needed to limit its wide spread use and balance the tension between sharing and controlling. Yet, other times, when there is a range of existent social and technical boundaries already there to regulate its movement and use, it is crossing these boundaries that demands extra work. In this respect, existent social or technical boundaries are conditioning, and manifested through software sharing.

Social boundaries between different teams play certain roles here. We've mentioned how homemade software was openly shared within the close-knit and interdependent research team. However, if it is across teams, things become a lot different. For instance, Huang's awareness of how his software was used and his responsibility for its use was largely confined to his team, "*we've been using it internally, and I don't quite know about other teams.*" We also noted that extra resources (e.g. social capital, social negotiation, and financial charge, academic credit) were needed for software to travel across teams, especially for homemade and unpublished software. Bai, for instance, when asked whether they were allowed to use Huang's software (they were from the different teams) suggested social capital was important, "*we are (allowed), because he is a former student of our director.*" Other times, social negotiation were required, as Liu described explicitly, "*yeah, in theory, it is fine, but he belongs to another lab after all, so if we want to use his software tool, we need to go ask for his permission, and he might need to get permission from his lab director.*" She further suggested that the negotiation process was important as a way to express respect to the boundary, "*usually, if you go ask him, he is nice and is willing to give, but a negotiation process is needed after all.*" It suggests that, what matters here is the negotiation itself and the attitude of respect to the boundary expressed through asking.

The purchase and shared use of commercial software are further shaped by and manifest the existence of social boundaries. In our study, the purchase decision of commercial software was always made at the team level,

never individual. On one hand, the cost of commercial software was too high for individuals to afford. On the other hand, as one informant pointed out, even if they paid for commercial software, it could not guarantee to solve the problem, as research was essentially explorative after all. So it was rare for individuals to purchase software, and all reported that they would try the best to find free software when possible. Once the commercial software was purchased, its use was usually freely shared within the team, as Bai put it, "*usually if it is lab owned software, all members of the lab can use.*" Others outside the team but in the same institution could use it, but they were of secondary priority in terms of its use and were usually not allowed to control it. Liu described main and marginal roles of different teams, "*After it was purchased, it was mainly our team who used it. Other teams could if they wanted…but it was mainly our team who used it, and that would be fine.*" Essentially, the difference between ownership and use of software marks the boundary between inside and outside the team.

Interestingly, when the software is super-expensive, or requires certain expertise to use, the boundary becomes more visible. Qiangrui described, "*If it is within our own institute, just let them use. Unless the software is very very expensive, and then we might charge (for fees to use it).*" We did encounter cases that the software was expensive, and it took special expertise to use it. In that case, others could only use the software through services provided by the team, as described by Qiangrui, "*we help them identify protein with our techniques, and we go through the whole procedure by ourselves, without their involvement, so they are not able to access the software directly.*" They also charged certain fees based on how much time it took to use the software to do analysis, "*yeah, yeah, if it takes longer to output the data, the fee will be higher, and in contrast, if it is some simple identification work, it will be lower.*" Other times, it took academic collaborations or academic credit to use software across the boundary, *"usually it is because when there are many samples to work on. Considering the funding constraints, then they will (choose this approach). It is a kind of compromise. That is, I don't need to pay that much by letting them co-author our paper."* It also suggests that the effect of social boundaries on software sharing depends on expertise required and effort needed to use the software.

Besides social boundaries, software sharing may also be regulated and managed along technical boundaries. For example, the intranet provides a natural boundary for managing and controlling the use of commercial software. The use of EndNote is a case in point, as described by Qiangrui, "*Because almost everyone will use it, our institute purchased it, and you could download it for free within the whole institute's IP addresses. That is, our institute has paid for the copyright. It is restricted by the IP address, and you can't use it beyond it.*" Interestingly, sometimes, the scope of an intranet itself can be leveraged to manage the sharing tension for work-in-progress software, as shown in Bao's case, "*It is only used in the intranet for testing, and it becomes public when it is published.*" Therefore, these boundaries are not there simply to negatively constrain the spread of software, but to positively resolve various tensions involved in software sharing.

DISCUSSION

In this paper, we have shown how scientific software sharing is conditioned by a range of technical and social factors, similar to the sharing of other scientific resources. Particularly, we illustrate how different types of software are shared within different social arrangements, according to different circulation mechanisms, associated with different social meanings, and subject to different boundary regulations. Specifically, homemade scripts and work-in-progress software are mainly circulated within one's own or related teams, shared based on membership, through personal requests or exchanges, and regulated largely along the internal and external boundary. Published academic software is made publicly available and is circulated mainly within the research community, relying on paper publication and citation mechanisms for its awareness and selection, and calling for peripheral participation practices for new members to acquire effective access to it. Commercial software products are developed, released and purchased mainly at the level of institution or team, where social-technical boundaries are leveraged and contrived to regulate its flow. Open source software comprises a great portion of scientific software, and its circulation is subject to the norms of open source software licenses. Its use in science also relies on peripheral participation practices for new members to navigate through it effectively. As such, the sharing of software is conditioned upon the interaction between the characteristics of software and its corresponding social configurations.

Software can be conceptualized as being in between digital data and physical instruments. As with digital data, it can be easily copied and distributed through IT infrastructures, and as with physical instruments, its key lies in the execution of certain tasks. To a certain extent, software, as a set of computer programs, essentially is an instrumental tool in the digital form. Of course, data, instruments and software are not separate, but many times are interrelated. Data is usually collected by instruments, and stored and managed with software. As a matter of fact, when our informants talked about software, often they meant database. The database may not be the direct subject of their scientific analysis, but forms an important means for the analysis of other data. Also, software can be part of the physical instrument, and subject to physical constraints for being moved.

However, it is the different situations and the different roles software plays in science that fundamentally transforms what is critical for the effective flow of software. Previous

studies of scientific sharing revealed that scientists carefully maintain good control of their resources and are reluctant to share so that they can exchange for some other benefits, such as authorship, funding and others [8]. However, the issues of concealment and reluctance did not appear very salient for software sharing in our study. Arguably, two factors contribute to this difference. One, since a wide range of open source software and published academic software is already widely available, it is hardly a case that people gain greater priority than others by simply possessing a piece of software. Two, while critical, software might not be as central or as uniquely valuable as the data for scientific discovery, especially when the tools are not directly producing data. Consequently, the critical issue of software sharing transitions from issues of simply making the resource available to issues of awareness, selection and navigation. Just as when more and more information is available online, what becomes problematic is not the availability of information itself, but how to locate, navigate through, and select relevant and high quality information. In our study, we found awareness gained internally through group interactions as well as externally through publications, paper reviewing, citation, and experience sharing have all become important means for software location, selection and navigation. While awareness, selection or navigation issues are also relevant to the sharing of other digital resources such as data, they appear to be more pronounced in software sharing, mainly because the bar for software production is much lower than that of data production. While only scientists or related staff in the domain can produce data or operate the instrument, far more people can use their computers to produce software, which makes issues of quality control, software selection, awareness, or navigation more salient.

Nonetheless, not all software in science can move freely without resistance. Issues of resistance and strategies for boundary management are still quite relevant, especially for work-in-progress software or commercial software. In our study, we uncover various tensions between sharing and controlling, and people even intentionally contrive socio-technical means to set boundaries and resistance to its flow. We believe it is the inherent value associated with software that makes the resistance and boundaries necessary. Although it can be freely copied, essentially, software is not value free, as its production involves labor, profession and innovation. The value may be realized through reputation economy or financial economy, and boundaries are essential for these systems to work. Here, what makes software different from data is probably that it takes longer to be produced and mature. As shown in our study, many times, the software is continuously being improved over time, and users' involvement and feedback is crucial in this process, which makes the tension between sharing and controlling even more salient. Yet for digital resources, such as software and data that are less bound with physical instruments, managing boundaries becomes more difficult and complex. In our study, we show that not only are existing technical and social boundaries leveraged to resist the free flow of commercial software products or work-in-progress academic software, but also people strategically contrive socio-technical mechanisms to manage its flow such as binding the use of software to physical machines by generating machine related code, choosing different media for sharing (physical copy, email or web), and so on.

CONCLUSIONS AND IMPLICATIONS

By focusing on software as an object of sharing, not as part of the infrastructure or site for sharing, we do not limit ourselves to sharing issues of crossing distances, disciplines, and organizations as did most studies of collaboratories [11, 44]. But rather we were able to take a close look at social practices of scientific software sharing both locally and remotely. In particular, with an bio-informatics team in China, we uncover different meanings and mechanisms of sharing within different social contexts – learning, apprenticeship and membership at the local, and papers, citations, and reputation at the communal, as well as a set of boundaries, existent or contrived, regulating the travel of software.

In particular, we draw a number of conclusions and implications based on the study, and hope to incorporate them into our future work in designing computing platforms to support relevant practices.

First, not only software itself but also the special situation of software, including the existence of open source software, the role it plays in the field (e.g. a central role in bio-informatics), as well as the local culture (e.g. Tidui in China), plays important roles in shaping how software is used and circulated and what is critical for its use and circulation.

Second, we uncover how mechanisms and meanings of scientific software sharing are embedded within different collective social practices, such as learning, apprenticeship and so on, which shall be carefully considered when designing relevant computing support for scientific practice. For example, considering learning is an important theme within a team, sharing source code directly might not always be desirable, and enhancing interactive learning experiences might be more important.

Third, as with more and more software made publicly available, and especially considering the more universal production and ubiquitous nature of software, we argue that what is important is not simply making more software available, but addressing issues of navigation, selection and awareness. For example, enhancing awareness of what each researcher or research group has done at the local scale could be highly valuable here.

Four, in a related vein, we argue that experience sharing is important for addressing issues of software selection and navigation, and that the request-and-give social protocol is

an appropriate means to turn tacit experience into sharable information. It suggests that computing enhancement to the sharing of experiences through the request-and-give protocol, rather than the sharing of software itself, might make a better fit for the corresponding social practices.

Finally, instead of forcing the sharing or the moving of software, respecting corresponding boundaries and seeking ways to support related social negotiation processes are also important, especially for work-in-progress software or commercial software products. After all, although software can be copied and distributed for free, it is essentially not "value free". Its value can be realized through either a reputation economy or financial economy, and boundaries are essential for these systems to work.

ACKNOWLEDGMENTS

We would like to thank our informants for their support, and Drew Paine for his editorial assistance. The work is supported by the National Natural Science Foundation of China (NSFC) under Grants No. 61272533 & No. 61233016, and the Shanghai Science & Technology Committee Key Fundamental Research Project under Grant No. 11JC1400800.

REFERENCE

1. Akhlaq, U. Impact of Software Comprehension in Software Maintenance and Evolution, *Blekinge Institute of Technology*, 2010.
2. Androutsellis-Theotokis, S. Open Source Software: A Survey from 10,000 Feet. *Foundations and Trends in Technology, Information and Operations Management*, 4, 3-4 (2010), 187-347.
3. Armbrust, M., Fox, A., Griffith, R., Joseph, A.D., Katz, R., Konwinski, A., Lee, G., Patterson, D., Rabkin, A. and Stoica, I. A view of cloud computing. *Communications of the ACM*, 53, 4 (2010), 50-58.
4. Atkinson, M., DeRoure, D., Dunlop, A., Fox, G., Henderson, P., Hey, T., Paton, N., Newhouse, S., Parastatidis, S. and Trefethen, A. Web Service Grids: an evolutionary approach. *Concurrency and Computation: Practice and Experience*, 17, 2‐4 (2005), 377-389.
5. Begole, J., Rosson, M.B. and Shaffer, C.A. Flexible collaboration transparency: supporting worker independence in replicated application-sharing systems. *TOCHI*, 6, 2 (1999), 95-132.
6. Bietz, M.J. and Lee, C.P. Collaboration in metagenomics: Sequence databases and the organization of scientific work. *ECSCW 2009*, (2009), 243-262.
7. Birnholtz, J.P. What does it mean to be an author? The intersection of credit, contribution, and collaboration in science. *Journal of the American Society for Information Science and Technology*, 57, 13 (2006), 1758-1770.
8. Birnholtz, J.P. and Bietz, M.J. Data at work: supporting sharing in science and engineering. *In Proc. GROUP'03*, ACM (2003), 339-348.
9. Bonaccorsi, A. and Rossi, C. Comparing motivations of individual programmers and firms to take part in the open source movement: From community to business. *Knowledge, Technology & Policy*, 18, 4 (2006), 40-64.
10. Bos, N., Zimmerman, A., Olson, J., Yew, J., Yerkie, J., Dahl, E. and Olson, G. From shared databases to communities of practice: A taxonomy of collaboratories. *Journal of Computer‐Mediated Communication*, 12, 2 (2007), 652-672.
11. Carlson, S. and Anderson, B. What Are Data? The Many Kinds of Data and Their Implications for Data Re‐Use. *Journal of Computer‐Mediated Communication*, 12, 2 (2007), 635-651.
12. Colazo, J. and Fang, Y. Impact of license choice on open source software development activity. *Journal of the American Society for Information Science and Technology*, 60, 5 (2009), 997-1011.
13. Crowston, K., Wei, K., Howison, J. and Wiggins, A. Free/Libre open-source software development: What we know and what we do not know. *ACM Computing Surveys (CSUR)*, 44, 2 (2012), 7.
14. Dourish, P. and Anderson, K. Collective information practice: Exploring privacy and security as social and cultural phenomena. *Human-Computer Interaction*, 21, 3 (2006), 319-342.
15. Evans, J.A. Industry collaboration, scientific sharing, and the dissemination of knowledge. *Social Studies of Science*, 40, 5 (2010), 757-791.
16. Fischer, B.A. and Zigmond, M.J. The essential nature of sharing in science. *Science and Engineering Ethics*, 16, 4 (2010), 783-799.
17. Foster, I. The grid: A new infrastructure for 21st century science. *Physics Today*, 55, 2 (2002), 51-63.
18. Gentleman, R.C., Carey, V.J., Bates, D.M., Bolstad, B., Dettling, M., Dudoit, S., Ellis, B., Gautier, L., Ge, Y. and Gentry, J. Bioconductor: open software development for computational biology and bioinformatics. *Genome Biology*, 5, 10 (2004), R80.
19. Hackett, E.J. Essential tensions. *Social Studies of Science*, 35, 5 (2005), 787-826.
20. Hey, T. and Trefethen, A.E. Cyberinfrastructure for e-Science. *Science*, 308, 5723 (2005), 817-821.
21. Howison, J. and Herbsleb, J. Socio-technical logics of correctness in the scientific software development ecosystem. *In Proc. CSCW'10 Workshop*, ACM (2010).
22. Howison, J. and Herbsleb, J.D. Scientific software production: incentives and collaboration. *In Proc. CSCW'11*, ACM (2011), 513-522.
23. Kelly, D. and Sanders, R. Assessing the quality of scientific software. *In Proc. Workshop on Software Engineering for Computational Science and Engineering* (2008).
24. Kelly, D.F. A software chasm: Software engineering and scientific computing. *Software, IEEE*, 24, 6 (2007), 119-120.

25. Kensing, F. and Blomberg, J. Participatory design: Issues and concerns. *Computer Supported Cooperative Work (CSCW)*, 7, 3 (1998), 167-185.
26. Kibrick, R., Conrad, A. and Perala, A. Through the far looking glass: Collaborative remote observing with the WM Keck Observatory. *Interactions*, 5, 3 (1998), 32-39.
27. Kling, R., McKim, G., Fortuna, J. and King, A. Scientific collaboratories as socio-technical interaction networks: a theoretical approach. *In Proc. Americas Conference on Information Systems (Long Beach, CA)*. (2000).
28. Knorr-Cetina, K. *Epistemic cultures: How the sciences make knowledge*. Harvard Univ Pr, 1999.
29. Lave, J. and Wenger, E. *Situated learning: Legitimate peripheral participation*. Cambridge Univ Pr, 1991.
30. Luscombe, N.M., Greenbaum, D. and Gerstein, M. What is bioinformatics? An introduction and overview. *Yearbook of Medical Informatics*, 1 (2001), 83-99.
31. Merton, R.K. *The sociology of science: Theoretical and empirical investigations*. University of Chicago Press, 1979.
32. MYERS, J.D. Tools for Collaboration. *Collaboratories: Improving Research Capabilities in Chemical and Biomedical Sciences: Proceedings of a Multi-Site Electronic Workshop*, National Academies Press (1999.
33. Nguyen-Hoan, L., Flint, S. and Sankaranarayana, R. A survey of scientific software development. *In Proc. ESEM'10*, ACM (2010), 12.
34. Olson, G.M., Atkins, D.E., Clauer, R., Finholt, T.A., Jahanian, F., Killeen, T.L., Prakash, A. and Weymouth, T. The upper atmospheric research collaboratory (uarc). *ACM Interactions*, 5, 3 (1998), 48-55.
35. Rosen, L. *Open source licensing: Software freedom and intellectual property law*. Prentice Hall PTR, 2004.
36. Rother, K., Potrzebowski, W., Puton, T., Rother, M., Wywial, E. and Bujnicki, J.M. A toolbox for developing bioinformatics software. *Briefings in bioinformatics*, (2011).
37. Sanders, R. and Kelly, D. Dealing with risk in scientific software development. *Software, IEEE*, 25, 4 (2008), 21-28.
38. Santos Jr, C.D. and Kon, F. ATTRACTIVENESS OF FREE AND OPEN SOURCE PROJECTS. *European Conference on Information Systems*, (2010).
39. Santos Jr, C.D., Cavalca, M.B., Kon, F., Singer, J., Ritter, V., Regina, D. and Tsujimoto, T. Intellectual property policy and attractiveness: a longitudinal study of free and open source software projects. *In Proc.CSCW'11*, ACM (2011), 705-708.
40. Scacchi, W. Free/open source software development: Recent research results and methods. *Advances In Computers*, 69 (2007), 243-295.
41. Schön, D.A. *The reflective practitioner: How professionals think in action*. Basic books, 1983.
42. Segal, J. Some problems of professional end user developers. *In Proc. VLHCC'07*, IEEE (2007), 111-118.
43. Segal, J. Models of scientific software development. *In Proc. Workshop on SECSE'08* (2008).
44. Shrum, W., Genuth, J. and Chompalov, I. *Structures of scientific collaboration*. Cambridge: MIT Press, 2007.
45. Singh, P.V. and Phelps, C. Determinants of open source software license choice: A social influence perspective. *Available at SSRN:http://ssrn.com/abstract=1436153.* (2009).
46. Strauss, A.L. and Corbin, J. *Basics of qualitative research*. Sage Newbury Park, CA, 1990.
47. Sun, C., Xia, S., Sun, D., Chen, D., Shen, H. and Cai, W. Transparent adaptation of single-user applications for multi-user real-time collaboration. *TOCHI*, 13, 4 (2006), 531-582.
48. Vertesi, J. and Dourish, P. The value of data: considering the context of production in data economies. *In Proc. CSCW'11*, ACM (2011), 533-542.
49. Wenger, E. *Communities of practice: Learning, meaning, and identity*. Cambridge Univ Pr, 1998.
50. Willinsky, J. The access principle: The case for open access to research and scholarship. Cambridge, MA: MIT Press, (2006).