Tracing Transnational Flows of IT Knowledge Through Open Exchange of Software Development Know-How

Susan Elliott Sim
Department of Informatics
University of California, Irvine
Irvine, CA 92697-3440
+1 949 824 2373
ses@ics.uci.edu

Kavita Philip
Department of Women's Studies
University of California, Irvine
Irvine, CA 92697-2655
+1 949 824 7092
kphilip@uci.edu

ABSTRACT
Information technology (IT) is often promoted as a socially and culturally agnostic tool that will allow emerging economies to leap into the digital age and reap the wealth that accompanies it. But in addition to the programming language, software tools, and books, *know-how* is needed to turn bright ideas into innovative, marketable solutions. This know-how can only be acquired from experience or from other IT developers. An effective means for sharing know-how is through an open exchange, which we characterize as a space where interested people can learn, critique, and contest ideas. It's a locale that is defined in terms of activity, rather than geography, technology, or membership. We use open exchanges of know-how as an analytical lens to example historical examples and contemporary instances. While conducting fieldwork in India, we observed open exchange occurring at Barcamp Bangalore. It is through these exchanges of know-how, rather than the transmission of tools or software artifacts that IT knowledge flows between international locales. We conclude this paper with a discussion that is mutually informed by contemporary practice and historical configurations.

Topics
Community technologies and networking
Information technology and services

Keywords
Open Source, software development, knowledge, know-how, transnational circuit

1. INTRODUCTION
Information technology (IT) is commonly touted as the route to national progress and the renewal of the global economy. While there are many good reasons to accept this claim at face value, we suggest that future IT innovation might better be served by analyzing this process (a) technologically and (b) socio-historically. A more nuanced formulation of this claim is likely to have beneficial effects on various kinds of policy-making that base themselves on versions of this claim.

First, we wish to disaggregate the notion of knowledge itself, and specifically IT knowledge, into two components, "know-what" and "know-how." No doubt IT could be further subdivided, but at a first level of approximation, this heuristic division reminds us that technology itself is both a knowledge-system and a practice. It is both a highly systematized, explicitly formulated and repeatedly tested set of logically nested truths (which allows us to *know what* laws hold), and a highly fluid set of practices, intuitive beliefs, and implicit codes (which allow us to *know how* to do things).

There has been much historical and sociological work on expertise (and "*know-what"* forms of IT knowledge), but less on the intuitive, socially coded set of practices we wish to lump under the heading of *know-how*. We suggest that understanding know-how will help us better understand some of the ways in which (1) groups of software developers share knowledge, (2) software development innovates, and, (3) IT knowledge travels along transnational circuits of practitioners.

Thus the policy areas that are elucidated include questions such as: Where should we look for the next wave of ideas in software development to fuel the nation's economy? We suggest in closing that many sites of often-overlooked creativity and innovation may exist in marginal networks outside the mainstream areas of software development.

Below, we briefly explain what we mean by know-how. We then explore why a more complex ethnographic understanding of know-how might elucidate the ways in which software innovation occurs, and how this shifts our models of global dispersion of technological practices. Finally, we explore the policy implications.

2. WHAT IS TECHNOLOGICAL KNOW-HOW?
In their classic work "The Social Shaping of Technology," Donald Mackenzie and Judy Wajcman [1] suggest that technology has three layers of meaning: physical objects, the human activities associated with these objects, and most importantly, knowledge about how to conceive, design, build and repair these objects, a rather fuzzy area they term know-how. This latter area, "know-

how," is fuzzy, and not merely to *post hoc* analysis. More intriguingly, expert technological practitioners often cannot put into words how they know what to do when confronted with a technological challenge. Other theorists have suggested that know-how cannot be captured in words, and is often visual and tactile, not just mathematical or verbal [1] (pp. 3-4), [2]. Historians of technology have suggested that this sort of implicit, practical skill at the nexus of art and craft is in fact the older meaning of technology itself, and have explored the variations of meaning in la technologie (French, "technology"), la technique (French, "technique"), and die Technik (German, "technics," in Lewis Mumford's translation).

A set of issues is raised for us by this STS approach to technology and the practical arts. First, how does know-how function in software development? Second, what can we learn from other historical examples? And finally, how does this function in a global, not just national, context?

2.1 The Role of Know-How in Software Development

Know-how has an important role in software development and its acquisition is necessary to attain proficiency in the craft. Whereas know-what is explicit, factual knowledge, know-how is the ability to put know-what into practice. Know-how is necessary to successfully innovate. Programming languages, application frameworks, and software tools are general-purpose technologies. They are intentionally designed to be highly flexible and adaptable, and the onus is on the developer to use these tools to create specific solutions. Furthermore, this know-how needs to be constantly updated, because software technology and the information ecologies of end-users are constantly changing.

Know-how is the link between creativity and innovation. Creativity can be described as the generation of ideas, while innovation is the practical application of those ideas into workable solutions. Know-how is the procedural and experiential knowledge that is needed to perform the transformation successfully. In information technology, know-how takes many forms, such as working knowledge of application program interfaces (APIs) and libraries, the craft skills for creating a database schema, and the judgment to know when to apply different principles. This IT know-how is highly situated, which means that it needs to be adapted to specific situations, and constantly emerging, because the problems and technologies are constantly changing. Therefore, open exchanges where people can seek out, provide, and share IT know-how are particularly important in this domain for sustaining innovation.

2.2 Open Source Software

The most commonly cited space of open exchange, is the domain of open source developers, who work through communities based on exchanging and increasing know-how. Open Source software is often cited as means for leveling inequalities between intellectual and material haves and have-nots. Although Open Source developers have forged ingenious modes of sharing knowledge, it is in fact surprisingly difficult to share know-how through software artifacts.

This difficulty is due to properties of source code. Program source is too complex to be understood on its own (or even with typical documentation), too brittle to travel well between settings, and contains completed solutions, but not the know-how needed to build new solutions. The constraints of the problem, the problem context, and the decisions made as part of the design process are missing. Many technologies are complex, but software especially so, and this complexity is often compounded by its size. An application like Microsoft Excel can contain upwards of ten million lines of source code. It can be very difficult to locate the know-how in this tangled mass of classes, interfaces, design patterns, delocalized plans, and scattered concerns. As well, it can be difficult to modify, since the omission of a single file, or even an error in a single line of code can cause a program to fail to compile into an executable that will run correctly.

Open Source software artifacts are necessary, but not sufficient to transmit know-how. Software development know-how needs to be acquired through experience or from other practitioners. Every successful Open Source project has a corresponding community that interacts through forums or mailing lists, and sometimes through embodied meetings. We have found that open exchanges are critical to the exchange and transmission of software development know-how.

3. OPEN EXCHANGES FOR SOFTWARE DEVELOPMENT KNOW-HOW

An open exchange is a space where interested people can learn, critique, and contest ideas. It is not so much a physical or virtual place, or even a particular event, but rather an opening for a particular kind of interaction. In this sense, the classical Greek agora, as a marketplace of ideas, a crossroads for intersecting contingencies, and a forum for a critical public community, was also an open exchange. It's a locale that is defined in terms of activity, rather than geography, technology, or membership. Another formulation of an open exchange in terms of activities is "link, lurk and try," meaning linking with others of like minds, lurking on the periphery of a community of practice, and trying out new things with low risk [3]. Examples of "trying" are accessing and using new technologies, airing new ideas, and rehearsing arguments.

Any site that has the capability for a community of individuals to interact with each other directly has the potential to become an open exchange. They are not necessarily geographic locations or regular events; they can be virtual spaces and informal meetings. Their memberships can be restricted and controlled, or they can be informal and unregulated. The exchanges can be public, or they can be within an organization. What distinguishes a merely social group from an exchange is common practice, that is, the organization of the community around a set of problems, technologies, or know-how. The adjective "open" applies to the sharing of ideas and solutions, unfettered by hierarchical structural constraints, reporting relationships, and professional rank. The most effective open exchanges appear to be ones with a diverse membership, with people representing a broad spectrum of local contingencies and social groups. John Seely Brown has argued that it is the trust and "creative abrasion" in such communities that is the key to innovation [4]. Innovation, he argues convincingly, cannot be "managed," but instead must be critically nurtured by creating a space for pluralism—neither stifling it, nor letting it run amuck.

The Open Source movement has resulted in a worldwide community of practice and a network of open exchanges [5, 6]. Communities of practice are a highly effective means for learning know-how, especially in domains involving design and

technology [7]. The project source code, discussions in electronic forums, and solutions in the form of bug tracking and change sets is open for anyone to examine, comment on, and contribute to. Transmission of know-how occurs through both active participation and legitimate peripheral participation, or lurking.

3.1 Historical Examples

There are many historical and contemporary examples of innovation that has been fueled by open exchanges.

Libraries are early examples of sharing. The great libraries of Alexandria, Tunis, Tibet, Nalanda, Baghdad were centers of shared learning to which scholars would travel, sometimes for years, to avail of free knowledge. A revolution in cataloguing during the 1800s revealed a dense web of connections among multiple knowledge elements and processes, "transforming the library catalog from an inventory into an instrument of discovery" [8]. By identifying these connections and innovative combinations of knowledge elements, the library became an open exchange and fostered knowledge creation.

Open exchanges have followed global transnational circuits long before the modern era. For centuries, preachers, traders, warriors and adventurers carried shared experiences, ideas, and memories around the globe, creating global markets and shaping networked histories [9]. While some analyses have characterized these flows as traveling between cores and periphery, the story is a more complicated one than the standard one about conquest, domination, submission, and tribute. Rather, transnational circuits of commerce and culture became the conduits for the later development of sharing, hybrid networks. Today, open exchanges for know-how within transnational circuits are central to innovation in software development.

4. Open Exchanges of Software Know-How in India Today

Around the world, BarCamps are *ad hoc* gatherings of software developers, explicitly formulated as an open, interactive exchange [10]. Forged in opposition to perceived exclusions in the sharing of software development know-how, BarCamps, often referred to as "non-conferences," challenge the hierarchies among speakers and audience, keynotes and panels, experts and laypersons. Organized in the form of collectives that meet on the fly, BarCamps exhibit a mode of *ad hoc* community exchange that employs the most flexible current tools of know-how exchange, including wikis, wifi, social bookmarking, photosharing, blogging, and chat. They have become one of the most popular "semi-official" ways in which software developers learn from, and forge, communities of practice.

BarCamps all over India have been vital in projects such as localization of software, popularizing new programming languages and techniques among non-native English speakers, and the discussion of the social and political context of emerging IT economies. One of us (Philip) attended BarCamp Bangalore (BCB) [11] in August 2007 to learn about open exchanges in this context.

BarCamps are full of a palpable excitement. Participants give up weekend leisure (after grueling work-week schedules in corporate programming jobs) in order to meet with people they come to consider their most intimate community, yet whom they largely know only on-line. The combination of virtual and physical worlds, technical and social discussions, work and leisure, and multiple programming and human languages makes BarCamps thrilling examples of open exchange for participants. One of the main organizers of BCB expressed his commitment to open public exchanges of knowledge as the primary reason for his devotion to the BarCamp project.

Although BCB participants are not identical to the Open Source community, many of them spoke of their involvement in India's free software movement. Many BCB participants reported searching for open exchange forums not primarily from ideological opposition to proprietary software, or social commitments to transparency, but simply because they found they were not learning rapidly enough in closed systems.

The emergence of open exchanges such as BCB serve to underline our critique that the transmission of software artifacts through the Open Source movement is not sufficient to transmit know-how. As well, social networking practices and technologies are not sufficient to create the communities of practice that are necessary to locate and share know-how. Rather, it is the configuration of the elements into open exchanges that is necessary to create an autonomous cadre of software developers that is needed to develop the software needed by local contexts. In this manner, IT knowledge flows across national boundaries to bring information works and countries into a global context.

5. Historical Lessons, Revisited

Is today's Open Source revolution another manifestation of the parameters that have historically governed technological progress?

Our preliminary investigations suggest that, although the current period of IT innovation shows many novel aspects, there are strong historical resemblances, which are worth investigating further. For example, historian Pamela Long's study of "Technical Arts and the Culture of Knowledge from Antiquity to the Renaissance" [12] finds that in the fifteenth- and sixteenth-century, technological practitioners commonly advocated openness, freely disseminating their knowledge of subjects such as mining, ore processing, artillery, and fortification. Other historians of science have shown that early modern scientific communities were rooted in "moral economies," formed by webs of values about collaboration, self-discipline, and sharing [1, 13]. These values were initially drawn from the ambient culture, such as the model of the humble, dedicated, and self-disciplined saint, which influenced seventeenth century models of the natural philosopher, and remained intrinsic to eighteenth century scientific personality-ideals. These values, however, morphed over time into specific scientific values without direct correspondences to the broader culture.

Scientific cultures reworked and re-circulated a dynamic web of values to form moral economies particular to each scientific community. The resulting scientific "moral economy" could not be explained simply as a reflection of cultural norms. More recent ethnographic studies of "hacker" communities suggests that their technological practice is closely tied to their "cultural" values regarding freedom, individuality, sharing, and innovation. For example, ethnographer Gabriella Coleman suggests that the form and content of Open Source software embodies structures of linkage, transparency and connectivity that are dynamically related with similar values in the Open Source community [14]. Since Open Source communities are globally dispersed, and function without conventional face-to-face interaction, there are few social mechanisms to enforce structures of exchange. Rather

than cultural or organizational conventions, it is technology that functions as the medium of exchange, embodying "values" of openness.

The mutual embeddedness of open cultures of exchange and technologies of open design is only just beginning to be understood. We suggest that the combination of historical, ethnographic, and software engineering methodologies provides a promising route to a robust understanding of this important dynamic, which is a part of the most innovative new software design practices today.

Techniques and practices of sharing are some of the most creative, and the most controversial, technological developments of the last decade. It is the shared excitement about technological challenges that facilitates, and motivates, new kinds of remote communication. It is the speed and global scope of the Internet that allows young "geeks" to build global shared communities, and any perceived threat to widespread access fuels much of their activism [15-18].

Why do software sharing communities, or what we call know-how agoras, spend so much time and energy developing technologies and practices of sharing? Many Open Source developers have day-jobs in proprietary software companies, but spend their personal, unpaid hours coding and de-bugging software that belongs to nobody, yet is potentially anybody's. A shared excitement about dispersed problem solving, as well as a commitment to open exchange of IT knowledge, fuels Open Source communities. Compartmentalization (into departments and companies) appears to be wasteful (in terns of optimizing creativity), because knowledge is "siloed," meaning there is a narrow division of labor. Developers spend a lot (too much) of their time locating resources, and not enough time playing with the resources. Theorists across disciplines, including legal anthropologist Rosemary Coombe [19], ethnographer of globalization Arjun Appadurai [20], and scholar-entrepreneur John Seely Brown [4] have pointed out that information has a cultural and social life – that is, it is produced, shaped, exchanged, and designed in fundamentally social contexts, not just in individual minds. Sharing is a key necessity for this transformation in the patterns of design innovation, and the emergence of open exchange systems for software know-how is a key to creativity in IT.

6. Discussion and Implications

Histories of science and technology teach us how to look for webs of values particular to extremely innovative technical communities. These histories also indicate why an interdisciplinary method is necessary: while historians of culture and ethnographers might be alert to forms of communication, patterns of collaboration, and innovative organizational behavior, software engineers and hands-on programmers are needed to identify forms of creativity that are tied to the techniques of writing code, designing system architectures, and ensuring interoperability. Multiple methodological skills must be combined to discover what connections exist between technical creativity and cultural practices.

Consider the question of: Where should we look for the next wave of ideas in software development to fuel the nation's economy? We have suggested that creativity and innovation exists in marginal networks outside the mainstream areas of software development.

Corporations are rarely the source of radical new ideas; novel ideas typically come from cottage industries and grass roots organizations. A few examples from the Internet era are blogging, digital music sharing, and Open Source software. These are practices that originated at the margins of society, but have become mainstream, and subsequently adopted by corporations.

Venture capitalists have long understood that the next big thing can come from unlikely corners, so they are willing to take big risks on unknown upstarts because the payoffs can be huge. This observation is consistent with two decades of scholarship in the history of technology, which suggests not only that knowledge often flows from margins to center, but also that the very definition of central or universal knowledge is one that continuously incorporates margins and gives rise to new peripheries. A continuous dynamic emerges: intellectual and popular, mainstream and marginal, core and periphery, interact and shape each other in a historical spiral whose parameters are simultaneously social, political, and technical. Therefore, our hypothesis suggests that more work is needed in the understanding of sub-cultures at the margins (economic, social, and geographic). We should move to investigating margins as not simply "lacking" in the resources of the center but rather as potential sources of radical ideas.

7. REFERENCES

[1] D. A. MacKenzie and J. Wajcman, *The Social Shaping of Technology, Second Edition*: Open University Press, 1999.

[2] E. S. Ferguson, "The Mind's Eye: Nonverbal Thought in Technology," *Science,* vol. 197, pp. 827-836, 16 August 1977.

[3] J. S. Brown and P. Duguid, "Organizing Knowledge," *California Management Review,* vol. 40, pp. 90-111, 1998.

[4] J. S. Brown and P. Duguid, *The Social Life of Information*: Harvard Business School Press, 2002.

[5] R. Goldman and R. P. Gabriel, *Innovation Happens Elsewhere: Open Source as a Business Strategy*: Morgan Kaufmann, 2005.

[6] J. Lave and E. Wenger, *Situated Learning: Legitimate Peripheral Participation*: Cambridge University Press, 1991.

[7] G. Fischer, "Social Creativity: Turning Barriers into Opportunities for Collaborative Design," in *Eighth Conference on Participatory Design*, Toronto, ON, Canada, 2004, pp. 152-161.

[8] M. Block, "Library: An Unquiet History: A Review," in *Ex Libris*, 2003.

[9] N. Chanda, *Bound Together: How Traders, Preachers, Adventurers, and Warriors Shaped Globalization*: Yale University Press, 2007.

[10] "BarCamp wiki - The password is c4mp," 2007.

[11] "Main Page - BarcampBangalore," 2007.

[12] P. Long, *Openness, Secrecy, Authorship: Technical Arts and the Culture of Knowledge from Antiquity to the Renaissance*: The Johns Hopkins University Press, 2001.

[13] L. Daston, "The Moral Economy of Science," in *Osiris, Second Series, Constructing Knowledge in the History of Science.* vol. 10, 1995, pp. 2-24.

[14] G. Coleman, "The Political Agnosticism of Free and Open Source Software and the Inadvertent Politics of Contrast," *Anthropological Quarterly,* vol. 77, pp. 507-519, Summer 2004.

[15] Creative Commons, "Creative Commons." vol. 2008.

[16] EFF, "Electronic Frontier Foundation | Defending Freedom in the Digital World." vol. 2008.

[17] FSF, "Year end appeal - help protect your freedom! - Free Software Foundation." vol. 2008.

[18] Linux Kernel Organization Inc., "The Linux Kernel Archives." vol. 2008.

[19] R. J. Coombe, *The Cultural Life of Intellectual Properties: Authorship, Appropriation, and the Low*: Duke University Press, 1998.

[20] A. Appadurai, *The Social Life of Things: Commodities in Cultural Perspective*: Cambridge University Press, 1988.