Learning/Organizing in Linux: A Study of the '*Spaces in Between*'

Maha Shaikh
London School of Economics and Political Science
Dept of Information Systems, Houghton Street
London WC2A 2AE
m.i.shaikh@lse.ac.uk

Tony Cornford
London School of Economics and Political Science
Dept of Information Systems, Houghton Street
London WC2A 2AE
t.cornford@lse.ac.uk

ABSTRACT
We assume that open source communities or collectives are somewhat organized, we also assume that such collectives are capable of learning, and indeed do learn. However, it is far more difficult to say exactly where, when and how such learning occurs, or resulting (re-)organizing happens. Drawing on Clegg et al's [1] concept of learning and becoming this paper seeks to show, through a case study of the Linux discussion around version control software, how learning and organizing occur. The paper discusses the Linux community's engagement with BitKeeper and explains aspects of its adoption. In this we address version control software as not merely a collaborative, organizing vehicle but as a part of a generative duality.

Keywords
Organizing, learning, becoming, version control software, Linux.

1. INTRODUCTION
> *"To learn is to disorganize and increase variety. To organize is to forget and reduce variety".*
> (Weick and Westley, 1996)

Open source communities are intriguing collectives that seem to challenge our existing knowledge both of organizational forms and of organizing as a ubiquitous activity in human affairs. We can choose to see such collectives as a reified form of organization, one that seems to thrive on organizational learning and live in constant flux – a novel organizational form that seems to work. However, in this paper it is not the organization as such, the static frameworks or functional divisions, that we consider, but rather the performative activity of organizing that it engages in – the verb rather than the noun. To understand organizing in such a way clarifies that an organization only exists 'in its duration' and throws into question the idea that it can be considered as an 'ontologically stable object' [1]. Thus, as we consider change within an open source project such as Linux, we recognize that organizations do not for the most part undergo planned or managed moments of change - unfreeze, change and then a refreeze - but rather are always in motion. In order to study learning and organizing we need to somehow track this motion. In this paper we use a specific framing of the concept of learning drawing on Clegg et al's concept of learning and becoming as 'two mutually implicating ways of exploring and simultaneously constituting the phenomenon of organization'. We seek to show, through our case study of the discussion around version control software in the Linux collective, how these concepts can help us account for the complex unfolding of what might at first sight seem a simple and obvious issue, the need for and use of some version control software within this collective.

We then draw some tentative conclusions about the nature of the learning and organizing that occurs as this software actor is encountered. We see version control software as not just an expression of the desire to organize (to structure, control etc.), though it evidently is, but as a vector of learning which takes the Linux community beyond a current state and which generates momentum to move it elsewhere - the notion of becoming.

The paper outline is as follows, section 2 describes the concepts of learning, becoming and organizing as proposed by Clegg et al [1]. Then, using data from our case study, in section 3 we take one small case study from our larger study, the use of a merge algorithm to consolidate patches, and explore how the twin forces of organizing and learning pay out. Finally we draw out some implications of our work and conclude the paper.

2. ORGANIZING/LEARNING/BECOMING
Clegg et al [1] seek to reframe our understanding of learning and organizing through the concept of becoming, 'as a process through which an organization exists'. These three concepts, of learning, organizing and becoming, are essentially performative and intertwined working together. In this approach, organizing is seen as the movement to stability, formality and structure, while learning is the movement into or towards the unknown, to chaos or instability. Too much of either cannot be good for the long term prospects of any organization or collective but too little may be equally fatal. Thus, in the tension between the dimension of organizing (closing down discussion, resolving issues, managing information flows), and that of learning (opening up discussion, engaging with new issues, adsorbing variety and encountering new information sources), we can seek the generative processes that sustain and mutate a collective over time. The concept of becoming is then a way of renaming the grey area where learning and organizing pull in their different directions and shape the future of the collective.

Open Source Application Spaces: Fifth Workshop on Open Source Software Engineering (5-WOSSE) May 17, 2005, St Louis, MO, USA.

2.1 Organizing and Learning

Organizing, rather than organization is the focus of this work since the word organization implies something stable and static, but even maintaining any status quo requires change and adaptation [2]. To be an organization (or as we might state it a social and technical collective) such as an open source project, is to have a constant engagement with change. We cannot understand what an organization is unless we are clear as to the nature of this fundamental and on-going ability to organize and to be organized, the preconditions for which are some form of 'chaos, disorder and noise' out of which, like the primeval swamp, organization (life?) can emerge [1]. From then on every little breakdown or stimulus potentially forces an organizing response, which itself presupposes some exploratory and appreciative activity (learning) is occurring. Both environmental variety and the heterogeneous elements within the collective offer chaos and disorder from which learning can emerge. The organizing response is one that juggles and re-organizes in order to exploit, isolate or attenuate such disturbances and maintain order. Chaos and disturbance demands adaptation and opens up the organization to future possibilities; it is in this uncertain and scary movement towards some future state, some not quite known future, that is Clegg at al's 'becoming' – not certain, not defined, but a journey into the chaos. Thus, according to Clegg et al [1], organizing, like learning, occurs in the spaces or folds between an extant order (that which is already organized) and chaos, and 'organization is the knot, the fold, where order and disorder meet. It is the very process of transgressing the boundaries between the old and the new, the stable and the unstable'.

In this unfolding learning is deeply implicated but organizational learning, in its conventional sense, has never been easy to define though many have tried [3, 4]. As Weick and Westley's [5] propose it, there is a quality of an oxymoron, in the traditional notion of 'organizational learning' as an integrated activity for which some normative guidance can be given, a thought crystallized in the short quote in the beginning of this paper and on which the discussion here is built. In contrast, in the approach used here we understand learning as essentially a form of *disorganization* - of challenge to organization by engaging with chaos and breakdown, of expanding the range of possibilities, and of facing up to variety.

Thus the more potential there is to interact with chaos, to maneuver and respond, the more it gives rise to a greater variety of possible paths that learning opens up. The greater the 'slack' within the organizing process, where slack denotes a flexibility, unallocated resources, discretion, and an ability to experiment and improvise, the more potential there is for learning to chart new routes [6]. Weick and Westley [5] add that learning, in its interactive nature, is carried out and exercised collectively through language, "language is both the tool and the repository of learning" (p446), which leads to their position that, "learning is not an inherent property of an individual or of an organization, but rather resides in the quality and the nature of the relationship between levels of consciousness within the individual, between individuals, and between the organization and the environment", a position echoed by Bateson and those who have written based on his work [7].

2.2 Becoming

Becoming, in Clegg *et al's* model, is perhaps a more complex and opaque concept than either learning or organizing. Becoming is not a specific state that is achieved, but the movement from 'then to now', or now to the future, and is embodied in the tension between outward looking and exploratory processes of learning, and cautious and considered acts of organizing. This duality has to coexist in order for either to operate. Clegg et al [1] describe this fine balance between learning and organizing as a 'generative dance on the edge of a volcano' linking with Weick and Westley's [5] observation that "to learn is to disorganize and increase variety. To organize is to forget and reduce variety".

In proposing the concept of becoming Clegg et al. [1] emphasize that the focus is on the resulting movement, not on *what* has moved or where it arrives (at best mere snap shots, moments in time); becoming is about travel and mutation rather than what has mutated. Taking this approach, in order to understand or explore in a deeper way learning we need to be aware of the movements - the becoming – that occurs in the 'spaces' between order and chaos.

3. LEARNING/ORGANIZING IN LINUX: THE CASE OF VERSION CONTROL

Version control software adoption has been a significant concern for Linux developers from almost the time of the project's inception, debated often in terms of the role it (might) play in organizing the collective [8, 9]. As a focus for this work it offers an interesting study in which forces of organizing interact with forces generated by chaos and breakdown, and the route taken (the becoming) is experienced as an ongoing movement, not a resolved issue. Thus we see at various points in the history of Linux that version control is used (appropriated) for the purpose of control and organization, but also as a focal point for engaging with chaos and breakdown - for learning.

Table 1 reflects such a generative dance, identifying a number of moves to organize that the Linux community has engaged in over the past 10 years, and counter posing them with various breakdowns or opportunities for learning. In general we can see in this history the need to juggle between an imperative of slack to encourage improvisation and to incorporate the power of participants, but equally, as the mass of developers grew and the code elaborated, for greater organization and stability. Each move towards stability, for example the early use of a discussion forum for communication (the LKML), can be identified as an engagement with disorder (hence learning) and to focus and accumulate the attention and feedback of a greater number of people some space (slack) in lieu of a conference room was needed.

In the table each example of organizing is matched directly with a crisis or breakdown, as can be read by following the table across horizontally. However, the generative dance that uncovers the becoming is more complex and can be traced indicatively through the arrows which indicate interaction and tension in either direction. Thus, the adoption of version control software in the form of BitKeeper (BK) in 2002 [10] was in part an act of organizing that responded to the crisis of patches being overwritten or ignored [learning]; this in turn led to another crisis, one that questioned the legal protection of the Linux code, as well as framing an ideological conflict between disparate and

conflicting licenses. As another example, the decision to have a parallel release strategy for Linux was an organizing response which evolved from a need of the community for improvisational slack as software evolved.

4. *BECOMING* UNDERSTOOD IN LINUX

The concept of becoming expresses movement as learning and organizing are engaged, what happens in the spaces in between. This section attempts to enter this space in the evolution of the Linux collective by addressing two quite narrow and specific incidents drawn from our larger study of version control within Linux. The primary data source we use is the Linux Kernel Mailing List (LKML), a data source that lends itself to such a study in that it provides a view of a dynamic and relational working out through language. The LKML is one main mode of communication of the developers and through our study of it we can enter into the activity of organizing and learning.

The two specific examples that we focus on are, first, the adoption of the version control software BitKeeper (BK), and in particular its claim to a superior merging algorithm (organizing) to enable more patches to be incorporated. The second example is the establishment of a gateway from BK to CVS, a response to a breakdown of trust within the community, linked to concerns as to the sustained purity of the GPL'd code as it becomes engaged with BK.

4.1 BK Merge Algorithm

BitKeeper was introduced to the Linux community in early 1999 [11] as 'an *Open Source* distributed revision control system which I claim is a substantial step forward from CVS' [italics added]. This was a troubled time for the Linux community which was suffering from tension between the leader Linus Torvalds and the greater Linux developer community. The stresses of coping with a rapidly growing project, with its heterogeneous mix of developers and software had led to Torvalds being overworked and unable to handle all the patches emailed to him. The growth in the number of developers and growth in software [bringing with it modularity concerns] were combining to create a chaotic environment for work.

Bitkeeper was an attempt to address these concerns, software specifically tailored for Linux and Torvalds, and written by Larry McVoy of the company Bitmover.. The merge algorithm in this software was a key selling point. It was claimed as a significant improvement on how CVS and other tools merge patches [12] and has since, with Torvalds' adoption of BK and endorsement, been somewhat proven true. As Torvalds has written, 'in the case of something like the Linux kernel tree.. you've got at least 20 actively developed concurrent trees with branches at different points. Trust me. CVS simple CANNOT do this. You need the full information. Give it up. BitKeeper is simply superior to CVS/SVN, and will stay that way indefinitely since most people don't seem to even understand _why_ it is superior' [13].

The main improvement [12] is the 'the success rate of the merge algorithm [which] is made possible by storing certain kinds of meta-data for each file that neither CVS nor diff and patch can store or generate'. The BK merge is also capable of doing a graphical *three-way* file merge and very importantly, has the ability to auto-merge, 'I avoid patch rejects, and can take advantage of the automatic BK merge features' [14].

If we see the creation of the merge algorithm as an act of learning, an engagement with breakdown and chaos, then its adoption, long and painful as it has been, reflect the attempt to translate (or inscribe) this learning within the community and to offer a new node for organizing aligned strongly with Torvalds. Earlier attempts to organize around such functionality had been resisted, and CVS [Concurrent Versions System] which was and is the most widely used version control software, was earlier refused by Torvalds because 'I'm afraid that I don't like the idea of having developers do their own updates in my kernel source tree. I know that's how others do it, and maybe I'm paranoid, but there really aren't that many people that I trust enough to give write permissions to the kernel tree' [15]. But in this case the learning that was embodied in the algorithm made VCS acceptable.

4.2 BK→CVS Gateway

The BK→CVS gateway offers another view of an organizing response drawing on conflict and chaos. Our point of entry is the amendment of BitKeeper's license [BKL] and the addition of a clause in October 2002 to restrict any developer that was engaged in developing, or selling, a product in competition with BK to use BK. "I noticed Larry [McVoy] recently changed the license on bk… this would seem to be a change which is not Open Source developer friendly" because it now says "this License is not available to You if You and/or your employer develop, produce, sell, and/or resell a product which contains substantially similar capabilities of the BitKeeper Software, or, in the reasonable opinion of BitMover, competes with the BitKeeper Software" [16]. The pressure that this issue generated was evident from the number of emails that were sent urging that something be changed in the BKL to make it less restrictive.

It became obvious from McVoy's reply that there was not going to be any further amendments in the BKL that would make BK use more 'open source developer friendly' which spurred Pavel Machek to initiate his own version control project, BitBucket, which in turn created its own discussion as to the legitimacy of such a project.

McVoy's reply to Machek began a long set of emails, most poking fun at BK and McVoy's adherence to proprietary software. Issues which grew out of this discussion, and which reflect an engagement with disorder, were: that the developer base was getting diluted by initiating too many version control projects and thus not creating any truly useful ones, what Garzik called the 'SourceForge Syndrome' [17]; how a real conversation is needed before initiating any new tool (here McVoy having dinner with Torvalds and others was brought up as an example, 'Apparently Linus, DaveM and Larry did just this, 2 years ago and offline. bk is result of that discussion … [18] [19]; and why projects should be encouraged to keep open source the free and open infrastructure it is supposed to be, 'We should stop developing Linux now and remove all source code from kernel.org because it's reimplementation and further development of the APIs of the proprietary "UNIX" operating system is just a waste of time [20].

This suggests a number of opportunities for learning and responding emerged from this incident. We can see this movement via the developers discussing their concerns, their perceptions of breakdown, and their sense of what Linux was, should, or should not, become. One outcome (organizing) was, a few days later, McVoy announced that, 'We've been working on a

gateway between BitKeeper and CVS to provide the revision history' to users of version software other than BK [21].

5. IMPLICATIONS AND CONCLUSION

This paper is a brief attempt to understand how organizing emerges in the Linux collective through the discussion on version control software. The examples that we have used above are fragmentary and drawn out of a far richer reality of 10 years of Linux history. Still, we believe that they can help us to start to see some essential aspects of how this collective expresses its need to organize *and* to learn, to increase its variety and to stabilize its present. We also see here technology as a key element in this becoming, seen as an appropriate mediator and enactment of the alternative worlds that Linux may enter. For example, a world of hybrid licenses and grey metadata; a world of an empowered senior developer; or a world of one displaced by a smart algorithm. The version control software debate on the LKML gives a very strong sense of the duality of learning and organizing that express the becoming, and makes it very apparent that a tool that is at first glance so obviously adapted for help in organizing, leads rather to an engagement with the future of the collective and how it can promote and encompass variety through learning. Following these discussions on the LKML makes becoming a less opaque concept as the generative interplay of organizing and learning materialize through the dialogue of becoming.

This paper is a preliminary account, and reflects early work with this model, still we can draw some initial conclusions that we see as motivating our ongoing study in the area. First, technology as an actor is implicated in becoming. It is not only humans, or other organizational units or structures that are capable of movements. This leads us to explore not only technology's more obvious role as an organizer, but also its role as a learner through mutation, practices of use and appropriated functionality.

Second, we are interested in applying the concept of slack as a necessary prerequisite for learning, and how slack is achieved within the open source collective and to what extent a strong technology of organizing, represented by VCS, might drive down such slack.

Finally, this study suggest for us, that becoming operates on a number of planes and, for example, the ideological and license issues revealed are as tricky, as disordered and as challenging as the issues of patch merges, and will offer as many possible futures too.

6. REFERENCES

[1] S. Clegg, M. Kornberger, and C. Rhodes, "Learning/Becoming/Organizing," *Organization*, vol. 12, pp. 147-167, 2005.

[2] G. Bateson, *Steps to an ecology of mind*. New York: Chandler, 1972.

[3] C. Argyris and D. A. Schon, *Organizational Learning: A Theory of Action Perspective*. M A: Addison-Wesley, 1978.

[4] J. G. March and J. P. Olsen, "The Uncertainty of the Past: Organizational Learning under Ambiguity," *European Journal of Political Research*, vol. 3, pp. 147-171, 1975.

[5] K. E. Weick and F. Westley, "Organizational Learning: Affirming an Oxymoron," in *Handbook of Organization Studies*, S. Clegg, C. Hardy, and W. Nord, Eds. London: Sage, 1996, pp. 440-458.

[6] J. R. Galbraith, "Organization Design: An Information Processing View," *Interfaces*, vol. 4, pp. 28-36, 1974.

[7] S. Star and K. Ruhleder, "Steps toward an Ecology of Infrastructure: Design, Access for Large Information Space," *Information System Research*, vol. 7, pp. 111-134, 1996.

[8] M. Shaikh and T. Cornford, "Version Management Tools: CVS to BK in the Linux Kernel," presented at 25th International Conference on Software Engineering - Taking Stock of the Bazaar: The 3rd Workshop on Open Source Software Engineering, Portland, Oregon, 2003.

[9] M. Shaikh and T. Cornford, "Version Control Software for Knowledge Sharing, Innovation and Learning in OS," presented at Open Source Software Movements and Communities Workshop hosted by the International Conference on Communities and Technologies, Amsterdam, The Netherlands, 2003.

[10] L. Torvalds, "linux-2.5.4-pre1 - bitkeeper testing," University of Indiana, 2002 - Tue 5th Feb.

[11] L. McVoy, "revision control for the kernel (BitKeeper)," vol. 2004: University of Indiana, 1999 - Sun, 21st Feb.

[12] V. Henson and J. Garzik, "BitKeeper for Kernel Developers," presented at Ottawa Linux Symposium, Ottawa, Ontario Canada, 2002.

[13] L. Torvalds, "Re: BitBucket: GPL-ed KitBeeper clone," vol. 2004: University of Indiana, 2003 - Fri 7th March.

[14] L. Torvalds, "Re: BitBucket: GPL-ed *notrademarkhere* clone," vol. 2004: University of Indiana, 2003 - Tue 4th March.

[15] L. Torvalds and C. Schlenter, "Re: CVS, Linus, and us," vol. 2004: University of Indiana, 1995, 1996.

[16] T. Gall, "New BK License Problem?," vol. 2004: University of Indiana, 2002- Fri 4th Oct.

[17] J. Garzik, "Re: BitBucket: GPL-ed KitBeeper clone," vol. 2004: University of Indiana, 2003 - Sat 1st March.

[18] P. Machek, "Re: BitBucket: GPL-ed KitBeeper clone," vol. 2004: University of Indiana, 2003 - Mon 3rd March.

[19] J. Bradford, "Re: BitBucket: GPL-ed KitBeeper clone," vol. 2004: University of Indiana, 2003 - Sun 2nd March.

[20] C. Hellwig, "Re: BitBucket: GPL-ed KitBeeper clone," vol. 2004: University of Indiana, 2003 - Sat 1st March.

[21] L. McVoy, "Re: [ANNOUNCE] BK->CVS (real time mirror)," vol. 2004: University of Indiana, 2003 - Tue 11th March.

Table 1. The generative dance of learning and organizing through becoming.