Re-working by the Linux Kernel developers

February 10, 2003

Matt Ratto
Ph.D. Candidate
Department of Communication
University of California, San Diego
mratto@ucsd.edu

Preface
Technology design is generally a matter of re-working existing systems rather than the designing of entirely novel artifacts. In this paper I explore part of a computer operating system called Linux that is designed to be re-worked by its users, a process I call 'designing for redesign'. I examine the practices of reworking within this development effort using some concepts gleaned from activity theory, a meta-theoretical model that particularly focuses on the simultaneously material and conceptual aspects of artifacts. This work is two-fold; first to examine design as part of a larger activity of re-working, and second, to begin to put together a model of socio-technical activity that incorporates the complex epistemological and ontological conditions that characterize current human conditions. Understanding the sociality and materiality of "knowing" and "doing" in technologized society means unpacking what we mean when we talk of 'access' and understanding 'use' as often an activity of re-working.

Linux
Linux is a software program that can facilitate the operation of computer hardware. This type of program, called an operating system, provides a layer of abstraction between the computer hardware and the application software that the computer user actually uses to accomplish tasks. Without an OS, each piece of application software would have to directly address the internal workings of the computer, an almost herculean task given the mutiplicity of differences inherent in even the most similar of computer systems. More specifically, Linux is the kernel of an operating system made up of a number of different software programs that together accomplish the functions of an OS. While the kernel of an OS is only part of the overall code necessary to support a computer, it is, by far, the most complex and necessary part. While an OS might function with many of its pieces missing, it cannot function without a kernel.

It is important to note that while the label ‘Linux’ more appropriately refers to the kernel software that resulted from the development effort started by Linus Torvalds, it has come to include the many different collections of software that together with the Linux kernel come to form a complete OS. These collections, assembled and distributed by a number of different profit and non-profit organizations are collectively called ‘Linux distributions.' In this paper I focus on the Linux kernel development effort itself. And while there are many ways in which the kernel developers work can be understood as driven by some of the same issues that engage other developers within the overall Linux programming effort[1], there are also dissimilarities.

Activity Theory

Exploring change as similarly material and conceptual is one of the strengths of an activity-theoretical perspective. Further, such a perspective reminds us of the importance of material and cultural history on the emergence of novel material and conceptual artifacts. While activity-theoretical positions are far from monolithic (Cole, 1996 pg.139), there are some general similarities. First, activity theoretical positions share an interest in the simultaneously productive and communicative nature of social behavior.(Rossi-Landi, 1983) Second, activity theory shares with other practice-based theories an emphasis on the dialetical character of human experience, seeing structure and agency as similarly determined and determining.(Lave & Chaiklin, 1993) Third, activity theory understand cognition as 'distributed', incorporating both individual human subjects, the built environment, and other people.[2] Such perspectives offer rich ground for examining the work of Linux developers by focusing on the productive and communicative nature of their practices and the artifacts that mediate their work.

Designing for re-design

Many researchers have noted the way Linux and other open code development efforts seem to hold out the possibility for a radical rethinking of the relationship between technology makers and technology users.

However, the open code developers I studied did not appear to struggle with the same usability issues that plague developers of more mainstream technologies. Rather than develop software for a wide and diverse audience of users, many open source developers create projects directed towards users like themselves - technically sophisticated and computer-saavy individuals from similar backgrounds. This blending of user and developer often resulted in software that, while technically sophisticated, remained difficult to use and problematic to learn for non-computer professionals.

Rather than designing for 'users', many Linux developers seem to be creating software programs and libraries inherently directed towards 're-designers' - other programmers with the skills and knowledge to re-work the programs for their own use. While designing for re-designers does create communities that blend making and using, the skills and knowledge required to participate in this community puts membership out of reach for many computer users.

[1] For example, the kernel developers, like other Linux development communities, are constantly balancing issues such as attracting new users to Linux through both technical and social engineering, managing for-profit and non-profit development, and negotiating the complexity of the social arrangements between the different development efforts.

[2] This last position is commonly referred to as 'distributed cognition', (see for example [Hutchins,1995]. For more on the relationship between activity theory and other theoretical models see [Cole, 1996] and (Engeström, Miettinen, & Punamäki, 1999).

In an early email to a group of similarly inclined developers, Linus Torvalds set out the initial agenda for the creation of Linux.

```
Do you pine for the nice days of minix-1.1, when men were
men and wrote their own device drivers? Are you without a
nice project and just dying to cut your teeth on a OS you
can try to modify for your needs? Are you finding it
frustrating when everything works on minix? No more all-
nighters to get a nifty program working? Then this post
might be just for you :-)[3]
```

Note the nostalia for an earlier day in computing history when all users 'wrote their own device drivers' (and obviously were men). Projects, modification, all-nighters - these types of activities and the knowledges required to partake in them, can be understood as characterizing early Linux work. Although it can be rightly argued that since the early days of development, Linux has become a more 'user-centric' system, the number of times the quote reproduced above is called forth by Linux developers (and social researchers) indicates the continuing importance of this characterization.

In this paper I address how the Linux kernel development effort encourages some users to become designers , to move from a passive to an active role in the construction of their own technological experience. In one sense, such a move entails these users taking on a different relationship to technology, a relationship that can be understood as re-working practice that encourages a critical, creative relationship to the technology involved, as well as the incorporation of different modes of work around that technology. Briefly put, workers become re-workers by incorporating technical and well as social knowledges.

Re-working

An important initial step is to better define what we mean when we talk about "re-working". One of the difficulties in thinking through questions of reworking (and design) is the lack of a specific shared vocabulary for talking about similarities and differences in these activities. "Design" as a concept is too large to serve to guide our work without some necessary codification. Equally, other terms that describe practices of reworking are also vague. For example, customizing a car can refer to complex work such as new engines and other mechanical systems, or it can refer to merely cosmetic changes brought upon by say, installing new seat covers. Both types of work are customization - the making of something general into something specific and unique. However, the extent and intent of the work may be understood as different. This difference is important to understand, in part because of the political ramifications of these changes. For example, while changing seat covers is open to every car owner who has the available time and money, installing a new engine, upgraded suspension, or a fancy transmission require knowledge resources that are not necessarily equally available. Terms such as tailoring, tinkering, and redesigning, although somewhat different in scope, have the same vagarity as to the content of the changes wrought.

[3] posted to info-mini@udel.edu; from:Linus Benedict Torvalds; subject: Free minix-like kernel sources for 386-AT

Terms of reworking

I intially decided to follow the prescriptions of ethnomethodology and look for the 'member's categories' which referred to the work of reworking. I did a series of keyword searches on the Linux Kernel developers list[4] looking for four words I found were often used by Linux developers to refer to practices of reworking. The table below shows the results of those searches:

Redesign*	406[5]
Tinker*	300
Custom*	1351
Tailor*	50

Of the terms, those related to customizing and tailoring were the most straight-forward to define. In most cases, their use indicated that the work was being done (or had been done) in order to satisfy some kind of individual requirement or need. Here are a few examples:

```
Re: [Off topic] Re: New Linux distribution - PSL
(name removed)
Thu, 11 Jun 1998 08:11:57 +0000
.... Hmmm, wonder if a common baseline would result in slow
deaths of some distributions? There are bound to be less
difference, after all.
Or perhaps distributions would _then_ be just what
distributions should be all about:
The customizing of a baseline to specific needs.

Re: Linux stifles innovation...
(name removed)
Date: Fri Feb 16 2001 - 07:44:28 EST
...I have heard about businesses that write open source
software on order. I.e. customer
pay for customizing an open source package the company
knows well, then they release the extensions too.
```

In other posts, developers wrote about creating 'custom' kernels, writing software for 'custom' boards, or creating 'custom' applications. Others wrote about the process of 'tailoring' a kernel to meet specific needs or that others needed to 'tailor' the code created by the author in order to make it work 'for you.' While the definition of these terms seems obvious,, at least in so far as their majority use on the list, what is less clear is what such practices entail. The terms are used interchangeably to relate the 'individual' quality of the result (or what needs to happen in order to achieve that result), but the involvement of tools, standards, and/or other people is not defined. Posts about tinkering and redesigning provided more of a handle on which to grasp the different practices. Although these posts

[4] list archived from 1995- at http://www.uwsg.iu.edu/hypermail/linux/kernel/index.html
[5] These number are approximate.

also use the terms somewhat interchangeably, I decided to use the alternate practices of 'tinkering' and 'redesigning' to focus this paper.

Looking at design scholarship

To bound a chart of reworking practices I chose two concepts which seem to circle the work being done on the list (and often design and technology studies in general); the bricoleur and the engineer. These terms as articulated in *The Savage Mind* (Levi-Strauss, 1966) have served as a starting point for scholars interested in theories of mind, cognition, and culture as well as those interested more particularly in the practices of art and technology. Needless to say, the terms have been used both reflexively and unreflexively. I say this at the beginning to validate my inclusion of these terms in this thought experiment, but more importantly to acknowledge the problematic aspect of the use of such broad dichotomies. That being said, I want to expand on these concepts, address their problematic aspects, and try and see how they might, albeit after some transformation, fit into the field of reworking I'm exploring.

Moving from identities to practices

In *The Savage Mind*, Levi-Strauss attempts to define differences of thought in different cultures as savage or domesticated. He tries to reformulate the questions set forth by Durkheim and Maus in *Primitive Classification* (Durkheim & Mauss, 1967) and his teacher Levi-Bruhl in *Primitive Mentality* (L? vy-Bruhl, 1978) and *How Natives Think* (L? vy-Bruhl, 1985).

A bricoleur is one who makes use of 'bricoles' - odds and ends of previous work, left over stuff, materials that are 'at hand', but unrelated to the current task. In his ethnography *Working Knowledge*, (Harper, 1992) Harper uses the term to explain Willie, a 'jack-of all trades' in upstate New York. Harper rightly acknowleges that Levi-Straus presents the bricoleur "...first as a thinker: considering, reconsidering, always with a view to what is available." (pg.74)   This is obviously a far cry from previous anthropoligical generalizations of so-called 'savage mentalities', such as those of Levy-Bruhl, Levi-Strauss' teacher. In *Primitive Mentality*, Levy-Bruhl speaks of the natives of Greenland:

> ...[their] mental processes are not independent of the material objects which induce them, and they come to an end as soon as their aim has been attained. They are never exercised on their own account, and that is why they do not seem to us to rise to the level of what we properly term "thought."(pg.22)

For most scholars of that time, thought was 'properly' understood to be thinking, abstracted from the material conditions that brought it about. Levi-Strauss, however, opened up this definition to include the thinking aspect of the bricoleur, which he saw as a reflexive actor, albeit carrying out a different sort of reflection from that of his other category, the engineer or scientist.

However much Levi-Strauss helped to redefine what was considered to be 'thinking', it remains that his categories, the bricoleur and the engineer, are still problematic. Goody locates the problem in Levi-Strauss adoption of the dichotomy of mind into 'savage' and 'domesticated,' the 'minds' that are equated with the bricoleur and the engineer respectively. While Goody recognizes that Levi-Strauss is attempting to create a specific historical basis for this dichotomy by positioning the issue as historical, i.e. 'savage' minds are indicative of the neolithic age, while domesticated ones are characteristic of modernity,[Goody, 1977 #1618 pg.5) he also notes that this distinction carries with it the same "we-they" implications of previous dichotomies. Although Levi-Strauss is trying to overcome the limitations of his teachers, he ends up recreating, as Goody says:

> ... the ethnocentric binarism enshrined in our own categories, of the crude division of the world into primitive and advanced, European and Non-European, simple and complex.(pg.8)

These dire warnings are useful reminders of the problems associated with the use of simple dichotomies to characterize the world. And yet there is a difference here that does require explanation. The problem with Levi-Strauss' dichotomy is the application of this separation to specific groups of people. Levi-Strauss is attempting to explain societal difference, and the changing modes of thought that seem to precede or accompany shifts that are generalized as 'civilization.' Goody is trying to accomplish a similar task. However, he goes about it somewhat differently.

Goody places the difference in terms of 'changes in communication' and attempts to analyze "the relation between means of communication and modes of thought." He examines the genre and forms that accompany written language - lists, formulae, prescriptions, and recipes - and sees in them the types of thinking that are typically understood as characterizing abstract thought. Drawing upon Scribner and Cole's landmark work, *The Psychology of Literacy*,(Scribner & Cole, 1981) Goody explores how the 'modes of thought' often associated with 'civilization' are generated not by biological or genetic difference, but by the practices associated with certain forms. This move away from characterizing groups of people to characterizing forms and practices overcomes some of the problems outlined above. Thus, rather than characterize the issues I want to address as the difference between a bricoleur and an engineer, I focus on the associated practices of tinkering and redesigning. This difference goes a long way towards reclaiming the separation between the two extremes defined by Levi-Straus.[6]

[6] By locating the difference in practice rather than 'in the person', I want to highlight the shifting that goes on in most real world situations. Although it has been quite the 'mode' recently to speak of shifting identity in order to overcome the limitations in static identity-based analyses of society (using such terms as the bricoleur, engineer, the teacher, the student, and the sex worker point to this type of work,) my feeling is that identity is both too fixed and too fluid to really provide explanatory force. Identity is both too fixed to explain the rapid shifting that exists when work is going on, and too fluid to point to a motivating intention. Placing the activity and the practice at the center of the analysis allows us to overcome these limitations while still making use of the descriptive categories.

Tinkering

Tinkering and redesigning serve as the boundaries of practices in which we are interested. What practices are thus characterizable as 'tinkering'? A quick overview of academic literature reveals some uses of the term, many drawn from Francois Jacob use of the term to refer to the incremental process of evolution.(Jacob, "Evolution and Tinkering", Science, 196:1161, 1977] Knorr, for example, uses the term to refer to practices in which scientists make incremental and ad-hoc changes in the material infrastructures they use to accomplish scientific goals, "Scientists pursue doable research by tinkering with local contingency." (Knorr, 1979) 'Doable' work as tinkering is emphasized in Fujimura's article "The construction of doable problems in cancer research" (Fujimura, 1987) And Norris says that "science is tinkery business"(Norris, 1993) Nutch lays out a list of 'modes' associated with tinkering. These include; 1) using objects designed for other purposes, 2) creating research equipment from bits and pieces found around the research site, 3) modifying available tools, instruments, and equipment for coping with specific emergencies or project contingency, and 4) saving time and money by constructing a needed piece of equipment rather than buying it through 'conventional channels'. (Nutch, 1996)

Two aspects of 'tinkering' stand out. First, that as a practice tinkering involves a specific relationship between people and objects mediated by 'immediacy' and contingency. Tinkering is the accomplishment of 'doable' work - doable in the moment, making use of existing, rather than distant material resources. In other words, tinkering is first and foremost a practice of using 'bricoles' (to return to Levi-Strauss for a moment) to accomplish accessible tasks. Tinkering engages the artifact to be reworked 'directly.'[7]

Redesigning

This 'direct enagagement' with the artifact stands in stark contrast to the typical definition of designing, understood as the practice associated with the engineer. In *Engineering and the Mind's Eye*, Ferguson traces the development of modern engineering practice.(Ferguson, 1992) He sees this development as a move from the 'direct design' of the artisan (read: bricoleur or tinkerer) to the 'designing by drawings' of the modern engineer. Ferguson is quick to point out that these are "differences of format rather than differences of conception.(Ferguson, 1992, Pg.5) He goes on:

> Usually, the 'big,' significant, governing decisions regarding an artisan's or an engineer's design have been made before the artisan picks up his tools or the engineer turns to his drawing board. These big decisions have to be made first so that there will be something to criticize and

[7] It is necessary to caveat the term 'directly'. I use it, despite its troublesome nature, it order to indicate a difference between tinkering work and designing work. As Leontev explores in *Activity and Consciousness*, the postulate of directness is untenable. All activity is, in a sense, mediated. Thus, tinkering work, just like designing work is mediated by a number of different aspects, which include, but are not limited to, mental models, norms, language, and many other standards and tools. A similar caveat should be applied to the concept of 'immediacy.'

> analyze. Thus, far from starting with elements and putting them together systematically to produce a finished design, both the artisan and the engineer start with visions of the complete machine, structure, or device. (Ibid)

By characterizing both artisanal and engineering work as differences in format rather than conception, Ferguson tries to move away from the problems that accompany Levi-Strauss' bricoleur, engineer dichotomy. By placing the difference within the material tools used to conceptualize design, rather than the conceptual tools used to materialize it, Ferguson does not end up 'primitivizing' the artisan. Just as the engineer analyzes and criticizes, so does the artisan. The difference in their work is linked to material difference rather than mental ability. Just as Goody understands abstract thinking as related to the forms of lists and procedures, Ferguson sees the work of designers as being in the forms of the model, the blueprint, and the drawing.

Therefore, while tinkering work is characterizable as direct engagement with the artifact to be reworked, redesigning (and design work more generally) can be understood as work in which engagement with the artifact is deferred in favor of other physical artifacts that 'stand-in' for the primary artifact. As noted above, these include design drawings, models, flowcharts, blueprints, and the like. While tinkerers may be making use of conceptual artifacts such as mental models, designers articulate these models as physical artifacts in and of themselves. Again, to return to Ferguson, these are "differences of format rather than conception." The secondary artifacts used in the practice of designing, afford different capabilities from purely conceptual models, from each other, as well as from the immediate or primary artifact. For example, the blueprint of a building allows transportation, circulation, and the 'testing' of changes via erasing and redrawing. A scale model of the same building allows a '3-D' view and makes understanding the physical layout more straightforward. Obviously, both of these secondary artifacts allow for uses that are quite different than the building itself or from mental models of the building.

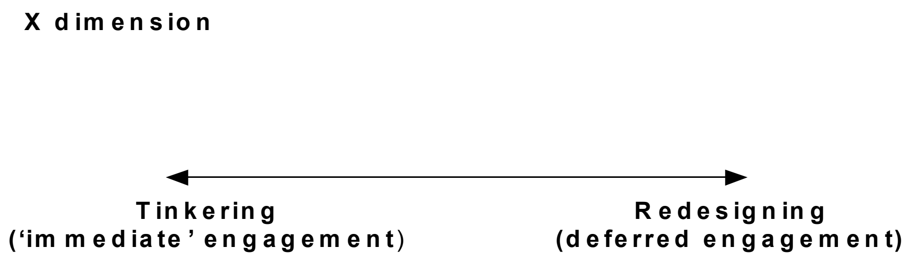
Primary, secondary, and tertiary artifacts

Above I use the terms 'primary' and 'secondary' to refer to the class of artifacts through which work takes place. I draw these concepts from the heirarchy of artifacts explored by Wartofsky(Wartofsky, 1979, pg.204) Briefly stated, Wartofsky defines three classes of artifacts: *primary artifacts* are those used directly in production and are typically thought of as physically existing[8]; *secondary artifacts* are understood as models or representations of primary artifacts, and are seen as providing for the transmission and preservation of modes of action and beliefs; *tertiary artifacts* are considered 'imaginative artifacts', things that engage us in a kind of "free play", and allow a re-imagining of current activity.

[8] It should be noted that although Wartofsky's examples included "axes,clubs, needles, and bowls" (Ibid) we might also include the material insubstantiation of things like words, either via sound vibrations or ink on paper.

X axis of chart

Without falling prey to a model of tinkering/redesigning based on mental ability, we can begin to articulate an important difference as being based on the artifacts involved in the work. This relationship is diagramed in figure 1:



(figure 1: picture of straight line described as 'reworking artifact', bounded on two sides by 'tinkering' and 'redesigning', left side is labeled 'immediate engagement', right side is labeled 'deferred engagement'.)

This diagram allows for more specific descriptions of the practices of reworking than the use of loosely defined terms such as customization and tailoring. Equally, tracing the different activities of reworking on this spectrum allows us to describe differences without recourse to identity dichotomies like bricoleur and engineer, dichotomies that seem to dissolve when actual practices are examined.

Secondary artifacts expanded

Above I noted the way designers use models, blueprints, and other secondary artifacts to 'stand in' for the primary artifact upon which work is being (or will be) done. It is important to include a further expansion of Wartofsky's notion of secondary artifact[9]. This is, namely, that secondary artifacts model not just the functional characteristics of the primary artifact, but other aspects as well. While a typical way to differentiate characteristics of technical artifacts is by separating these aspects into two classes, namely 'interface' and 'function', I see this binary as particularly problematic.[10] To

[9] For now I concentrate on the secondary level of Wartofsky's heirarchy. More work needs to be done to incorporate the tertiary artifact level. A good starting place for this work might be found by considering the notion of 'psuedo-concept' in Leontev, (Leontjev, 1978). Also, the idea of the 'imaginary' might be a rich location for further work. Some initial work has been done here including (Gregory, 2000) and (Hyysalo & Lehenkari, 2001).

[10] I explore this problem more specifically in 'Naturalizing Function' a paper delivered at 4S 2001. Briefly summarized, my argument is that the binary of 'interface' and 'function' can be understood as a corollary to the problematic dichotomy of 'social' and 'natural'. These binaries do similar work, locking off certain aspects of the world from criticism by opening up others. In this case, 'interface' is seen as 'social' and thus open to the kinds of understandings obtained via 'social analyses' while 'function' is understood as 'natural' and thus 'true', 'rational', and all the other characteristics typically applied to the natural world. That in the world of computer software such work has been institutionalized can be seen in the division of labor

overcome the problems of this binary, I replace it with three categories; conduct, function, and structure. These three categories loosely correspond to the different types of secondary artifacts used to model different aspects of the primary artifact.

To more clearly define these aspects of secondary artifacts, let me leave Linux behind for the relative simplicity of a different primary artifact, a styrofoam cup. The cup itself is the primary artifact, while a number of different secondary artifacts (that might be both conceptual and/or material) model the different aspects of the cup. For example, when you (as you drink your latte) remark on how styrofoam is not biodegradable and will probably end up in a land fill, the cup, or rather its representation be it verbal, mental, or material, has become a secondary artifact corresponding to its conduct - its durability and relationship to a physical world. Further, when you pick up the cup and note that the hot coffee it contains does not burn your hands, you are modeling the function (through action) of the cup - it contains fluid and does not transmit temperature. Finally, when you examine the cup, realize that it has small mold points on the sides and bottom, and hypothesize about the material conditions of its production in a factory, you are modeling the structure of the cup - how it was made and what kinds of organization was necessary in order to make it, distribute it, and use it. This example makes clear that secondary artifactualness consists in more than just physical models of how things work. They also include how things behave and what conditions are necessary for their production and use.[11]

Source code as multiple artifacts

This being said, what kinds of artifacts are used by the Linux kernel developers to 'stand in' for the linux kernel itself? One interesting aspect of computer programs is that their source code can be understood as both the primary artifacts upon which work takes place, as well as a model or plan of these primary artifacts. Thus. software makes obvious the way all artifacts work as both primary and secondary artifacts. (Miettinen, 1998) While it might be said that the Linux kernel only exists as a primary artifact after and only after it has been compiled, i.e. turned into object code, this seems a somewhat specious argument. Part of what makes software interesting as an object of study is that it makes obvious the way all physical artifacts live between this duality.

For example, legal scholars in the U.S. have hotly debated whether or not software is protectable as free speech, since it appears to have little difference from other formal languages[12]. In addition, intellectual property law has had a very difficult time

between 'human-computer interaction' and 'interface engineering' versus 'computer science' and 'software engineering.'

[11] There is an obvious connection here to alienation and Marx. Often capital production requires the transparency of these models by making them deliberately invisible or by replacing these actual secondary aspects with other, less politically or environmentally dubious models. This can be understood as part of the process of commodification.

[12] This work dates back to the 1960's and includes (Patent Resources Group & Bender, 1969), (Homet & United States. Congress. Office of Technology Assessment, 1983), (United States. Congress. Office of Technology Assessment, 1987), (United States. Congress. Office of Technology Assessment, 1990), and

characterizing software. In some cases, software seems to be expression and protectable by copyright law. At other times, software appears to be a machine, capable of carrying out actions upon a material world. In the US, machines can only be protected by patent or trade secret law. Again, some have tried to say that software as source code is expression and protectable by copyright and free speech law, while software as object code is only protectable as patent or trade secret and not open to the protections guaranteed by the US Bill of Rights (freedom of expression.) Many scholars have rejected these claims, noting that the distance between source code and object code is very small. An excellent example of the result of such deliberations was the outcome of a US court case against Phil Zimmerman, the creator of the encryption program PGP. Zimmerman, prevented from exporting PGP since all encryption software over a certain level of protection had been characterized by the US Department of Defense(of Trade?) as a munition, printed up the source code for PGP on a T-Shirt and attempted to board a plane for France. He was detained and the T-Shirt also was determined to be a munition under US law.[13]

The only way to resolve this issue is to give up and characterize software as both, both a machine and the expression of this machine, both the artifact on which work takes place and a model for this artifact. Giving up binary characterizations does not force us to give up difference. Instead, it requires us to do a very useful thing; to differentiate software by its use in practice. Thus we can say that sometimes the Linux source code is the primary artifact of work, sometimes it functions as a secondary artifact. The difference, as in all cases, exists in how it is being used at the moment of practice.

The shifting quality of artifacts is clearly described in Engestrom's chapter titled 'When is a tool' (Engestr? m, 1987). By rephrasing the question from 'what is a tool' to 'when is a tool', Engestrom gently critiques Wartofsky's heirarchical artifact levels by pointing to the constant shifting that goes on between them.

This seemingly simple move opens up vast possibilities for reconceptualizing the relations between signifier and signified, or to put it in other words 'word' and 'world.' The question thus becomes not what is the relationship between the word (or a sign or tool) and the corresponding physical world, but rather under what conditions and with what resources does the physical world become transformed into models (secondary artifacts) and back?[14]

For example, when programmers on the kernel developers list say that they are 'tinkering' with a section of the kernel code, we can understand that they are working directly on the

---

(United States. Congress. Office of Technology Assessment, 1992). More recent work includes (Samuelson & University of California, 1997), (Lessig, 1999), and [Burk, 1999]. For a more STS perspective on these issues see [Gillespe, 2001].

[13] There is an important point to be made here about the strategic control behind allowing for, or conversely preventing the transformation of things from primary to secondary artifacts. Looking for such strategies may permit a more sophisticated notion of 'access' - a concept sorely in need of redefinition.

[14] To rephrase Engestrom, it is not 'what is a word' but 'when?', and more importantly when is something 'not word.' Exploring these shifts is one way of moving beyond the 'linguistic turn' without returning to reductive forms of realism, and provides a methodological guide for those answering the recent call to 'practice' as the unit of analysis.

code itself, trying things out, adding syntax or changing algorithms in 'real-time'. Here are a few examples from the list itself:

```
Kernel 2.3.6 and ppa .... again
(name removed)
Thu, 17 Jun 1999 23:06:31 -0400
...I was tinkering with zipslack this time and attempted to
mount my slightly COD zip drive as umsdos. Fine. Did an
ls.....oops....seg fault....ls stuck in D state.

[RFC][DATA] re "ongoing vm suckage"
(name removed)
Date: Fri Aug 03 2001 - 18:44:43 EST
... IMO, far too much tinkering of code is going on
currently without hard data (other than "it looks good"),
and this is exacerbating the problems.

Re: Linux 2.4.10-pre11
(name removed)
Date: Tue Sep 18 2001 - 14:14:19 EST
The real question is why can't we just open 2.5 and only
fix the VM to start with? Leave the kernel at 2.4.1pre10
and possibly use the -ac VM code (which has diverged from
mainline),and leave people (Alan, Ben, Marcello, et. al.)
who want to tinker with it in small increments and do the
drastic stuff in 2.5.

ISS test3 is out
(name removed)
Wed, 5 Mar 1997 16:08:21 -0500
There are only two small behavioral bug reports I have
received at this point, one of them looks like a bug that
was in our networking before I began tinkering ;-)
```

Note that in these examples 'tinkering' and 'tinker' is always described in past or future tense: "I was tinkering, ...too much tinkering, ...who want to tinker, I began tinkering...". An initial insight is made clear by this form. Tinkering, at least within the Linux kernel developers list, is always based on practical, 'immediate', and material interests. Thus tinkering is only expressable on the list as something that happens outside it - before or after the expressive and denominational work that is being done on the list itself. The list itself, as a forum, is outside the bounds of tinkering. Another insight is thus possible; namely that the list itself forms one of the secondary artifacts by which designing and, importantly, redesigning practices take place. Here are some examples where 'redesigning' is being used:

```
Re: crypto (was Re: Congrats Marcelo,)
(name removed)
Date: Wed Feb 27 2002 - 17:29:47 EST
The one major downside, right now, is that Henry and
Richard et al, keep talking about redesigning the klips
structure to fit in with the more recent kernels better
(ala netfilter, maybe). They've announced some design specs
and I suspect that they would rather see the newer version
```

```
of klips in the kernel tree than the crufty version that we
are hobbled with in FreeSWAN right now.

Re: Journaling pointless with today's hard disks?
(name removed)
Date: Wed Nov 28 2001 - 13:46:24 EST
This is a last ditch deal-with-evil safety net system that
has a fairly good chance of saving the data without
extensively redesigning the whole system. Never said it was
perfect.

FAQ followup: changes in open fd/proc in 2.4.x?
(name removed)
Date: Fri Nov 17 2000 - 10:58:12 EST
Now we get to the reason for this post. Has anything
changed for 2.4.x? With release eminent, we don't really
want to go through the redesign and implementation if the
architecture is different for 2.4.x.
```

Note that in these examples redesigning is understood as a practice that involves terms such as 'specs', 'systems', and 'architectures'. Further, redesign is characterized as an ongoing practice, faciliated by conversation on the list. Thus, while tinkering is typically an 'off-list' process of one-on-one interaction with the work in question, redesign involves the mediation of secondary artifacts including the list itself.

Ideal developer steps

Based on my reading of the kernel list, I have compiled a series of typical developer 'steps'. These are:

1. Developer/user want to use a new network card on his Linux system
2. Developer needs to extend the Linux kernel to use this card.
3. Developer examines source code (primary artifact) for networking subsystem of kernel.
4. Developer tries making some changes to the code, recompiles the kernel as object code (primary artifact), and tests the new network card. He does this a number of times but it never works.
5. Developer posts a message to the kernel list, describing his attempts as 'tinkering', and posting a section of the original source code and his attempted changes. (conversion of code to secondary artifact)
6. Other developers give advice in the form of source code segments (secondary artifacts) as well as reasons for changes based on the overall structure of Linux (secondary artifact).

While this scenario might seem rather obvious (particularly to the developers themselves) two claims are defended here. First that source code serves as both primary artifact of work as well as a secondary artifact, and second, that the conversion of source code from primary to secondary artifact happens within the context of the list and other programmers. Before addressing this second aspect, let me call attention to the shifting of source code as artifact.

Shifting perspectives

Many scholars interested in design have commented on the perspectival differences between technology users and developers. (for example (Kling & Public Policy Research Organization (University of California, 1977), (Norman, 1990), (Carter, 1991) and (Johnson, 1998).) One example of such explorations can be seen in an article examining the development and subsequent abandonment of a automated post office kiosk called "postal buddy'. (Engestroem & Escalante, 1996) In this article, Engestrom and Escalante explore some of the ramifications of the differing perspectives between users and designers. One important finding was that while designers saw the technology in question (in this case, an automated postal services machine called 'postal buddy') as the object of their work, users saw the 'postal buddy' as a tool that mediated their real object of buying stamps or sending a package. The different objects of their respective activities made it difficult for designers to understand user perspectives which made the postal buddy difficult for users to use. Since in this case I'm talking particularly about a group of people who are both users and designers - in other words they share the perspectives and objects of both - it might initially seem that such a point is moot. However, as I hope to defend more particularly later in this paper, 'good' designs, (meaning useful, adoptable, and adaptable designs,) appear to emerge when designers are able to rapidly shift between the two object perspectives noted by Engestrom and Escalante. That shifting objectual practices is at the heart of expert work is a concept that only recently is beginning to be addressed.[15] In this paper my focus in on examining the different artifacts of reworking practices. This analysis is incomplete without a more detailed understanding of the connections between the artifacts that I examine and the objects of the activities in question. For now, let me focus on defining more particularly the chart of reworking in relation to the shifting uses of primary and secondary artifacts.

Y axis

One aspect of the shifting mentioned above is that it takes place on the list, within a community of other developers. My first response was to see the movement to the list as a movement from and individualistic process of work to a group process. But tinkering, like all human activity, is never fully and individual act. Symbolic interaction has pointed to the concept of the 'significant other' in order to capture this (Mead & Morris, 1974) and the notion of activity in activity theory itself is predicated upon historically (and thus socially) generated communities, norms, and rules. While it should be noted that in the latter tradition, the separation between 'practice' and 'activity' is based upon the extent to which the work being described is more or less social, more or less inherently within a community, maintaining this separation is often quite difficult. Becker (Becker, 1982) points to the difficulty in seeing any human work as singular. In one example, Becker

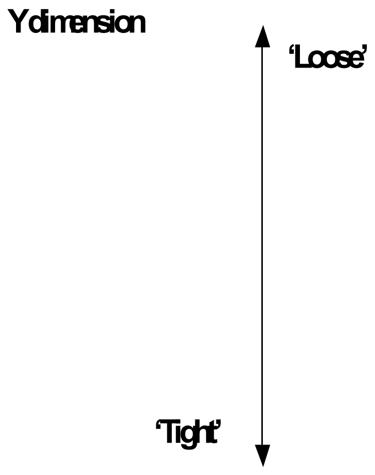
[15] A good example is the recent work of Knorr-Cetina on 'objectual practice' in (Knorr-Cetina, Savigny, & Schatzki, 2001). Also see (Turnbull, 2000))

demonstrates the sociality of Van Gogh's painting (a seemingly wholly individualistic act) by articulating the work of others to provide the paint, the brushes, and the canvas. In one sense, then, each brush stroke, each daub of color can be described as a social activity. Such a perspective is in line with my previous problematization of the notion of direct engagement. Just as work is never truly 'direct', it is also never only 'individual'.

That being said, how can we characterize the shift from tinkering sorts of work to the design work we want to describe? One useful distinction can be seen is the organizational 'style' of both practices. Tinkering seems to be a more tightly organized and stable practice, characterizable by a coherence of time and place. I don't mean to indicate that tinkering is organized from 'above' or that its rules are more explicit or formal. Instead, the organization of the practice of tinkering is often 'self-organized' and results in a coherent set of relations between the people involved, driven to be sure, by the simultaneous nature of the communications between then. In fact, tinkering work mostly takes place within and by groups of closely connected workers. Harper's description of Willie's shop is a good example of the 'connected-ness' of such practices. (Harper, 1992) Although these relations may change from location to location and from context to context, for the moment of the tinkering practice, they remain stable.

What the move to the list entails is thus a move away from the organizationally more 'tightly' connected tinkering work, and towards a more 'loosely' organized development effort. However, the term 'loose' should not be taken to mean that the rules, positions, or standards of the group are haphazardly decided or enforced. Instead, 'loose' indicates the spatial as well as a temporal distance between rules and enforcement, decision and adoption, tests and results. To use the metaphor of a rope, it is not that the knots are loosely tied, but that they are distantly spaced along the rope. Such spacing results is a sort of 'wiggle-room' that allows for different sets of problems - and different kinds of solutions - to arise within the groups involved.

These two terms, 'tight' and 'loose' have long been used by engineers to characterize technical systems. As Perrow points out, more recently the terms were adopted by some organization scholars working within the field of education, to indicate differences in social systems. (Perrow, 1999) However, Perrow uses the terms to refer to socio-technical systems - aggregations of people, communities, and artifacts. These terms can thus serve as the two points on our Y axis.



(figure 2: picture of vertical line labeled 'reworking organization'. Bottom of line is labeled 'tight', top of line is labeled 'loose'.)

The Y dimension describes another major difference between tinkering and designing practices. While tinkering can be understood as a practice more tightly organized by space and time, redesigning, particularly in view of the whole collective of Linux development, can be seen as a more loosely organized process. The 'loose-ness' of the development process is often revealed when people on the list talk about 'redesigning'.

```
Re: Synchronous board drivers
(name removed)
Mon, 5 Jul 1999 12:55:25 +1000
...That being so I'd like to run my current thoughts for
redesigning the ppp support in the linux kernel past people
on this list.

Re: Linux stifles innovation...
(name removed)
Date: Sat Feb 17 2001 - 15:05:36 EST
...I suppose you could argue that redesigning linux every
few years is innovation, but unfortunately its the same
cast of characters doing it, so its not very innovative.
```

As these examples show, redesigning is understood as a practice that requires certain kinds of consensus in order for its results to be accepted and used. However, often times, the need for building consensus in order to successfully carry out a redesign, is seen as a somewhat difficult process.

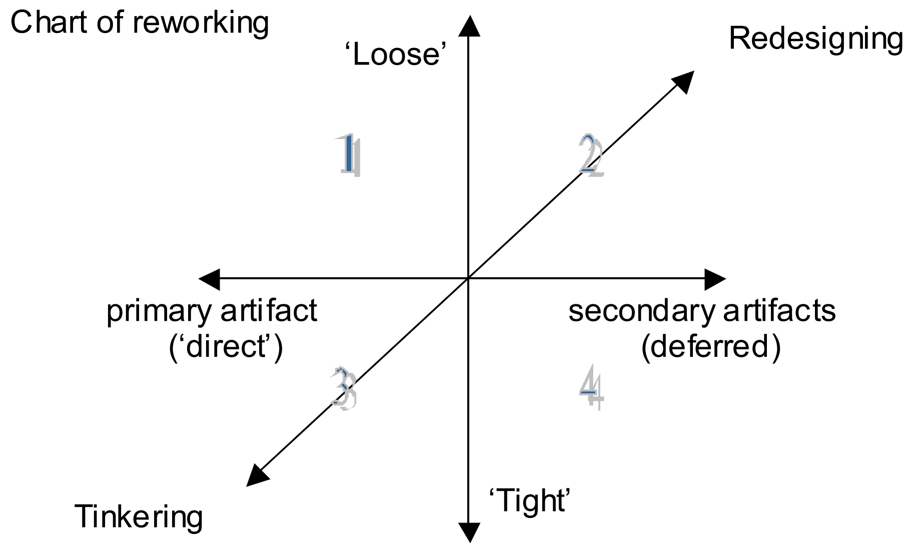
```
FAQ followup: changes in open fd/proc in 2.4.x?
(name removed)
Date: Fri Nov 17 2000 - 10:58:12 EST
So basically, before we begin the arduous task of redesign
is there anything in the 2.4.x kernel that will affect our
decisions?
```

```
Re: [PATCH] Re: Move of input drivers, some word needed
from you
(name removed)
Date: Tue Aug 22 2000 - 06:48:23 EST
My point is just that if we're going to go through the
horror of completely redesigning serial.c we might as well
get rid of the "char by char" mentality, at least not so
that it limits the _fast_ ports out there.
```

Note the use of terms like 'arduous' and 'horror' to describe the process of redesign. These terms are indicative of the way redesigning often works within the Linux kernel effort - by fits and by starts. While the loosely organized nature of redesigning in this community allows for certain kinds of freedoms, it also makes the global changes primarily driven by redesigning difficult to accomplish. Often times, such as in the last example, tinkering is understood as a way to accomplish changes without having to achieve the kinds of consensus necessary to successfully 'redesign'.[16]

Chart of reworking

Having laid out the X and Y axis of the chart. Let me put the whole thing together:



(figure 3: X and Y described as above, with diagonal line of tinkering/redesign)

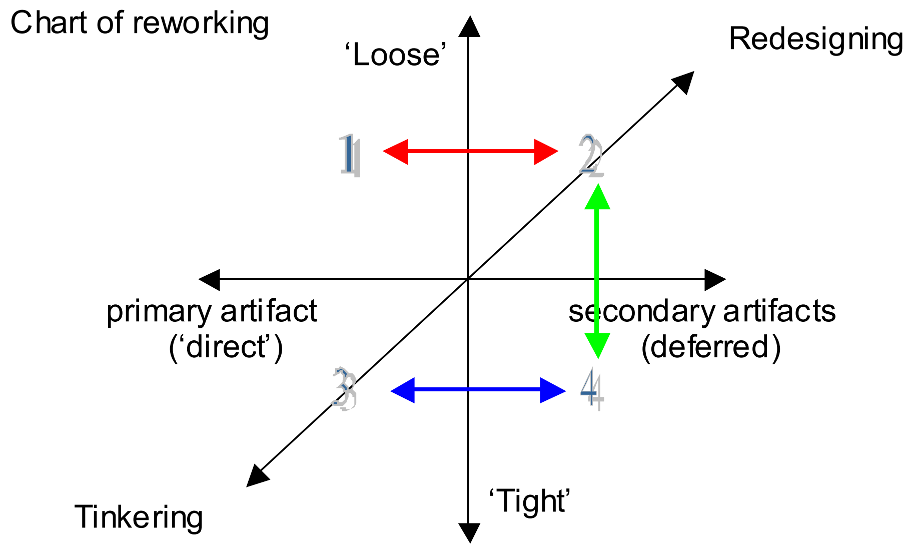
[16] A brief footnote here on consensus-building. Leigh Star has used the concept of 'boundary object' to critique the notion that science requires consensus building. 'Boundary objects' are considered to be artifacts which allow for communication to take place and work to continue despite disagreements as to the nature of the tasks at hand. Incorporating this idea into my analysis of the Linux effort would create a more nuanced way of talking about the kinds of consensus necessary to carry out the reworking practices I describe in this paper.

Note that with the addition of the vertical dimension, the activity of tinkering and the activity of redesigning now follow a diagonal path. Thus, tinkering can now be described as a 'tight' activity that involves 'direct' action of the primary object of work. Redesigning can be described as an activity involving 'loosely' organized people and work being done on multiple secondary artifacts which 'stand in' for work on a primary artifact.

While this may seem like a lot of work just to draw this fairly obvious relationship, there are some insights to be gained here. Let me start by revisiting the 6 reworking steps described above in a slightly different way.

3. Individual tries something
4. looks at source code
2. puts problem to list
1. multiple people on list try things
2. multiple posts to list
4. individual examines them
3. individual applies them to problem.

Redraw on the chart, this would look something like this.



(figure 4: diagram of shifts)

So what drives the shifts described above? One possible reason might be to overcome a lack of relational coherence between quadrants.[17] The initial shift between 3 & 4 might be driven by a gap (between quadrants 3 & 4) caused by the secondary artifact not adequately representing the primary artifact. In this case the primary artifact is understood as the future working code and the secondary artifact is incorrect source code.

[17] More conceptual work needs to be done here on this notion of 'relational coherence'. Two possible notions that might be useful starting points are the idea of 'gaps' drawn from Dewey, (Dewey, 1971) and the ideas of 'disturbance' and 'contradiction' from activity theory.

Equally this gap could be caused by an inability to see the relationship between the secondary artifact and the primary artifact. In this case both the primary and secondary artifacts are the source code and the individual needs help understanding why the code works the way it does. At any rate, the gap drives the shift to quadrant 2 - the incorporation of others into the problem. While the reworking problem might be solved at this stage, e.g. someone on the list immediately understands the problem and posts the solution, often times the other development list members themselves try out the code and post results (shifting between 1 and 2). This last shift can thus also be understood as driven by attempts to overcome a lack of coherence between quadrants 4 & 2.

Managing the shifts

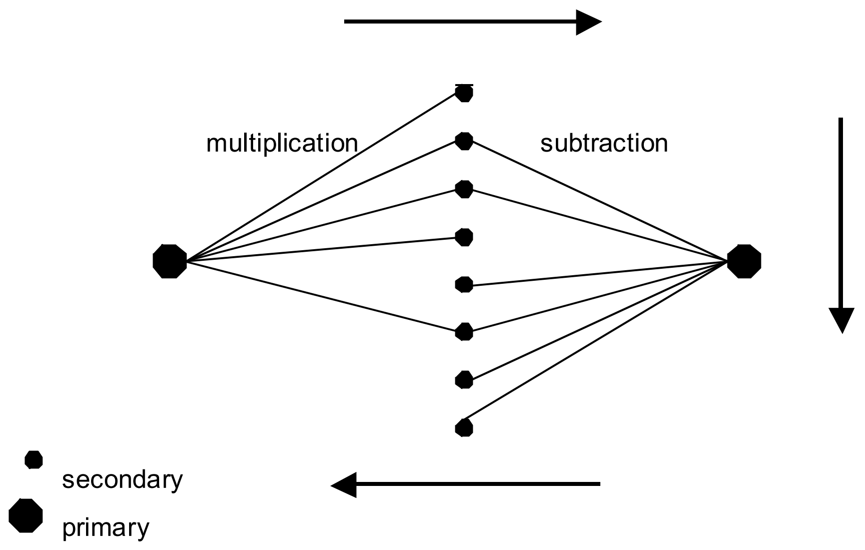
While this may seem a somewhat reductive way of understanding the activity of reworking, the above chart does reveal some important places to look when reworking practices are not successful. Gaps between quadrants and, more importantly, an inability to overcome these gaps by taking the work to another quadrant, can be understood as reasons for unsuccessful activities. There are many examples of these on the kernel list. The most obvious ones are when posts of problems are not picked up by other list members:

(example of custom kernel config post)

In this post, an individual asked for help creating a custom kernel. His attempt to make the shift to quadrant 3 was unsuccessful - there were no responses on the list. This is not to say that his reworking practice was ultimately unsuccessful. He may have received off list responses that would assist him. However, such responses would not indicate a successful move to quadrant 2, the incorporation of other list members and other secondary objects. Instead, responses would typically direct him away from the kernel list and towards other communities where he might be able to get assistance and incorporate other individuals and objects into his activity. In this example, it is easy to place the gap as being based on incomplete knowledge of the community rules and norms. The kernel list FAQ - itself an important secondary artifact - states that that questions about compiling and creating custom kernels will not be answered.

Artifact multiplication and subtraction

Just as upward moving shifts can restrict the success of reworking activity, downward moving shifts can also cause trouble. Earlier in this paper I articulated the expansion of secondary artifacts to include different aspects of the primary artifact, including conduct, functional, and structure and how these aspects of the artifact contain traces of uses, behaviors, and organization. But in addition to expansion, artifacts also multiply and substract through use. By multiplication and subtraction I mean that the models of primary artifacts (the secondary artifacts) increase and decrease as reworking goes on. To demonstrate this, let me diagram the process of reworking another way:



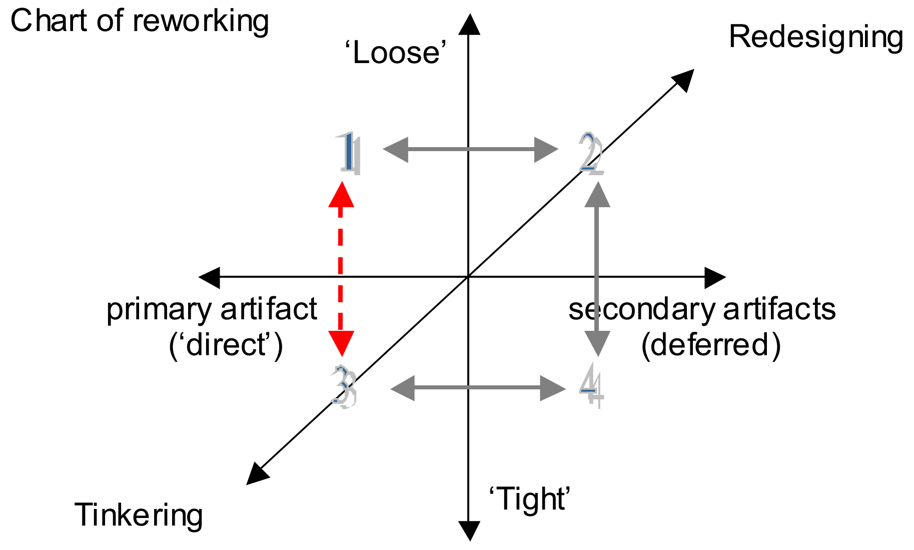
(figure 5 - artifactual change through reworking.)

In this diagram, the initial primary artifact (larger dot) is modeled by a series of secondary artifacts (smaller dots) some of which impact the primary artifact. Note that these secondary artifacts model may model the conduct, function, or structure (see above) of the primary artifact. Also note that the process of change detailed in this diagram is somewhat cyclical as represented by the two horizontal arrows.

This diagram makes it somewhat easier to see the multiplication of artifacts (shifting upwards) and subsequent reduction of artifacts (shifting downwards) necessary for a successful reworking process. It also reveals why it is often difficult to talk about success or failure. In many cases, the transformation of the primary artifact during the multiple shiftings between levels, results in a discontinuity between the beginning and end of the process. In other words, the mirror image of the multiplication of secondary artifacts is the multiplication of primary artifacts. That the circular motion depicted in the above diagram is often not circular at all is a reflection of the kind of circular quality of consciousness as explained by Leon'tev. Just as the 'subject-activity-object' cycle is broken in 'sensory-practical activity itself' (Leontjev, 1978, pg.78), the cycle of artifacts is also not a 'magic circle.' To think that the movement is purely cyclic is to believe two ridiculous notions. First, that the results of the multiplication of artifacts is wholly predictable, i.e. that there is always a rational progression from primary to primary artifacts, and two, that the resultant primary artifacts are fully realized (and realizable) versions of their secondary models. This latter mistake is (again) a clear indication of the problems with the postulate of directness. However, this should not be taken to mean that these movements have a tendency to stop. In fact, it often seems that stopping the multiplication of artifacts actually takes hard work.

A good example of this can be seen in Latour's examination of the 'black boxing' of two different artifacts, a computer system and a 'scientific fact.' (Latour, 1987) Stabilizing the Eclipse computer and the double-helix nature of DNA took immense leveraging of both institutional and 'natural' constructs. This stabilization can thus be understood as a process by which artifacts are prevented from multiplying via secondary artifact modeling. Equally, it might be possible for artifacts to achieve stability by its very lack - the subtraction that must take place in order for the changes explored via secondary artifact modeling to be reincorporated into the primary artifact. I take this as both a theoretical and a methodological point, first that artifacts 'just want to be free', i.e they want to become models for themselves and multiply; and second, that the stability of artifacts generated either through multiplication or subtraction requires explanation.[18]

Obviously, there are many other reasons why these shifts do not take place. Suffice it to say the the above charts provide places to start looking for reasons reworking activities are successful or unsuccessful, as well as a possible way of tracing the development of innovative aspects of the Linux kernel. Also, the chart of reworking makes visible an aspect of work on the list that is often ignored or forgotten. This is namely the lack of connecting line between quadrants 1 & 3.



(Figure 6 - return to figure 4 - no connection between 1 & 3)

It is difficult to see activities on the Linux kernel list within quadrant 1 - multiple people working together on the primary artifact. This is partially due to the form of the list - conversations on it take place in a spatially and temporally discontinuous or asynchronous mode - one aspect of the 'looseness' of the organization. That shared physical space is not a requirement is made obvious by the ongoing reworking practices that are facilitated by internet chatting, such as takes place on IRC and other syncronous electronic forms. However, that this quadrant is an important aspect of successful reworking activities is revealed by the many conferences, chat rooms, and shared work environments of the developers. Thus, the difficulties of connecting quadrant 1 to the rest of the chart provide us with a healthy reminder of the dangers of focusing too intently on

[18] Again, more work needs to be done on the multiplication and subtraction of artifacts that takes place around human activity. The recent focus on 'objectual practice' by Knorr-Cetina mentioned above is one starting point. Another is Cantwell-Smith's text *On the origin of objects*, (Smith, 1996).

just the interactions that take place within one form of communication, in this case the Linux kernel list.

Charting other 'reworkers'

How might other groups of reworkers be placed on the chart of reworking. Below is one attempt to place some particular groups.

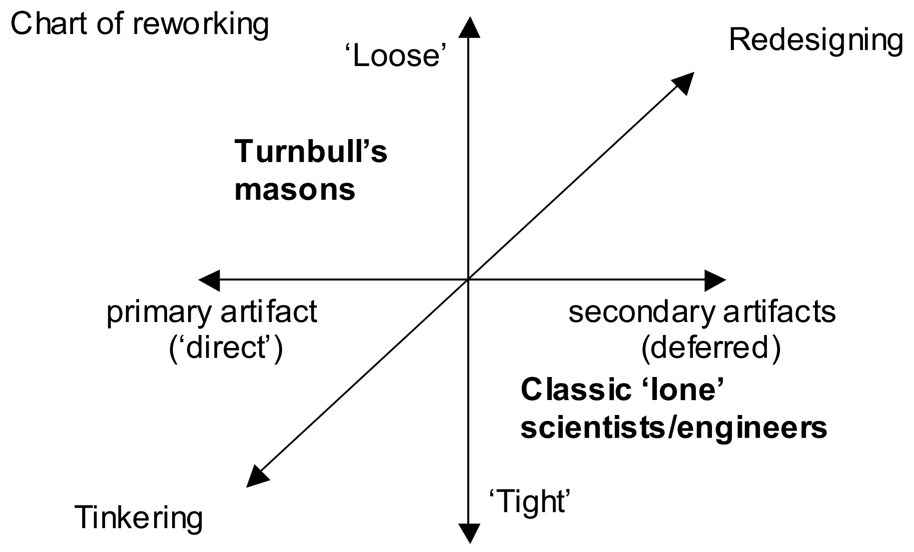


Chart of reworking
'Loose'
Redesigning
Turnbull's masons
primary artifact ('direct')
secondary artifacts (deferred)
Classic 'lone' scientists/engineers
Tinkering
'Tight'

(figure 8: Chart of reworking with other 'reworkers'.)

As you can see, the chart becomes another way of exploring views of working practice. In quadrant 1, I have placed the masons described in Turnbull's work on the construction of Chartres cathedral (Turnbull, 2000) As Turnbull notes, many architects and structural engineers held that Chartres, like other gothic cathedrals, was proof that architectural knowledge had been lost or great genius was at work in its construction. According to these opinions, gothic builders were working without the theoretic knowledge that is used today in structural analysis. This meant that explaining the construction of gothic cathedrals required "? the invocation of such imponderables as an 'insuperable barrier' or 'genius' or a long-forgotten secret building technique. (pg.55) Turnbull remarks that a more 'performative' approach overcomes the need to rely on such mysteries. Rather than great unknown genius architects with knowledges that have since been lost, the building of Chartres can be better understood as an 'experimental' activity, that brought together three essential components. First, talk, which allowed the patron (or his representatives) who planned the building and the masons who did the actual construction to collectively engage in an activity that melded the spiritual and scholastic traditions of the day with the necessary craft knowledge. Second, masonry traditions that provided continuity in technique as well as constant innovation. Third, templates, thin wood cut-outs that allowed masons to cut stones to the correct size in the absence of an overall plan. Thus, talk, traditions, and templates allowed for a kind of working practice that was both 'direct' in the sense that the work took place in local, physical ways, as well as 'loose' in the sense that the process took place without blueprints or architects (as we know them today), took place over hundreds of years, and was eventually completed despite stoppages that lasted for weeks, months, or even years. The success of such practices is obvious in that they created a building that has lasted over 800 years.

Turnbull's explication of cathedral building can provide some characteristics of 'loose' and 'direct' practices - use of templates rather than an overall plan, practices that move by 'fits and starts', directed by 'mastre masons' rather than architects. In fact, Turnbull's masons serve as a rather good parallel to the Linux kernel developers, especially when you consider the importance of ongoing talk (via the list), the way development traditions structure the work, and finally the way segments of source code of Linux serves as templates that guide overall development activity.[19]

In addition, I have positioned another category called 'lone classic scientists'. This work is characterized as 'deferred' and 'tight'. Such a characterization is a good reminder of the less than ideal nature of these categories. Just as most science work is neither purely 'tight' nor 'deferred' (see the articles by Norris, Nutch, and Knorr-Centina for good examples of this), none of the practices fit perfectly into these reductive categories. In fact, it might be said that most successful working or re-working activity relies on shifting between all sectors of the re-working chart.

Conclusion and future directions

First, it is important to note what is missing from the analysis above. First, in this analysis, groups of people working together are understood merely as aggregates of individuals. This does not give adequate attention to the notion of sociality as beyond mere aggregation. Social groupings possess their own characteristics, forces, and directions, and it is a mistake to understand them only as a sort of 'super-individual'. The key to understanding sociality is however, not only the incorporation of what might be considered to be the macro-societal issues faced by the sociality in question. These issues, which might include concepts of race, of ethnicity, of gender, and related questions of power, structure, and control, are important to address. However, 'going macro' is not the way to increase our understandings of these issues.

In part these difficulties are due to my focus on the artifacts of activity rather than the objects. A more detailed examination of reworking would have to address these connections more explicitly. The kind of analysis I've carried out in the above work, does little more than hint at the need for this more complete work. Thus, although the chart of reworking activity is an important starting place, the insights gained through such work can be made more relevant and more generalizable via a micro-transactional analysis.

[19] Such a parallel provides us a new way to understand Eric Raymond's work on open source software 'The Cathedral and the Bazaar'. Rather than the highly controlled and heirarchically organized process that cathedral building was supposed to be, it turns out to be much more like Linux. Over vast amounts of time (hundreds of years in some cases) a loose collection of masons and other builders built the Chartres cathedral. Instead of a complete architectural plan designed by an medieval engineer or architect, the masons relied on some simple drawings and physical templates in order to make such that all the pieces somehow 'fit' together.

Such work might start by looking at the kinds of resources required to carry out the activities noted above. These resources include the knowledges required to understand and write computer source code, access to computer and networking technology, the documents and shared history of the group, as well as the explicit and implicit rules and norms that structure the group experience and provide ways to negotiate, change, as well as revoke any aspect of the group dynamic. This latter dimension might well be called the political aspect of the Linux kernel work. Understanding this work as activity, as well as how this activity relates to the macro issues listed above and the meso-level understandings generated by my charting experiment, remains an elusive and important goal.

This paper is merely the beginning of a much larger project. And yet, a few useful insights have emerged from this work. The first point is methodological; one way to understand and analyze practice is by exploring the 'shifts' that go on in that practice. In the above analysis, I explored the shifting of Linus software development work by mapping the shifts on a chart made up of two dimensions; the x axis represented the artifacts used in the work, the y axis represented the kind of organizational 'coupling' that characterized different moments in the work. The analysis revealed that the ability to shift between the quadrants on the chart was essential to the ongoing nature of Linux development. Further, the analysis indicated that one reason for the necessity of shifting had to do with the coherence or discontinuity of the artifacts involved in the work. This latter aspect was indicative of the complex artifactual ground upon which the work took place. I briefly explored the multiplication and subtraction of the artifacts used in the work - i.e. the construction and destruction of this complex artifactual ground. The ability to navigate this ground can be understood as a version of 'expertise.' Thus, future work should take the following steps:

1) expanding the notion of artifacts to include the topological and 'ecological' relations between them - i.e. an 'ecology of artifacts',
2) expanding the notion of expertise to include experts who manage these relations.

Understanding how things multiply into secondary and tertiary artifacts and subtract into primary ones, as well as how they expand into objects and contract back into artifacts could thus serve as one way to reincorporate a notion of 'politics' into the study of technology and work. Starting from 'reworking' rather than working provides an important reminder not to focus too heavily on ideas of novelty and origins. The metaphor of 'ecology' provides one way to describe the emergent and shifting quality of technology reworking.

Bibliography

Carter, K. (1991). Participatory Design in a Commercial Environment, *CHI'91* (pp. 390), New Orleans, LA: ACM.

Becker, H.S. (1982). *Art worlds*. Berkeley: University of California Press.

Cole, M. (1996). *Cultural psychology : a once and future discipline*. Cambridge, Mass.: Belknap Press of Harvard University Press.

Dewey, J. (1971). *How we think; a restatement of the relation of reflective thinking to the educative process*. Chicago,: Regnery.

Durkheim, E., & Mauss, M. (1967). *Primitive classification* (Phoenix edition. ed.). Chicago: University of Chicago Press.

Engestroem, Y., & Escalante, V. (1996). Mundane tool or object of affection? The rise and fall of the Postal Buddy. In E. Bonnie A. Nardi (Ed.), *Context and consciousness: Activity theory and human-computer interaction.* (pp. xiii, 400): The MIT Press.

Engeström, Y. (1987). *Learning by Expanding*. Helsinki: Orienta-Konsultit.

Engeström, Y., Miettinen, R., & Punamäki, R.-L. (1999). Perspectives on Activity Theory. Cambridge: Cambridge University Press.

Ferguson, E.S. (1992). *Engineering and the mind's eye*. Cambridge, Mass. :: MIT Press.

Fujimura, J. (1987). The construction of doable problems in cancer research. *Social Studies of Science, 17*.

Gregory, J. (2000). *Sorcerer's apprentice : creating the electronic health record, re-inventing medical records and patient care.*

Harper, D.A. (1992). *Working knowledge : skill and community in a small shop.* Berkeley: University of California Press.

Homet, R.S., & United States. Congress. Office of Technology Assessment (1983). *The international dimension : new technologies and intellectual property rights.* Washington, D.C.?: Congressional Office of Technology Assessment.

Hyysalo, S., & Lehenkari, J. (2001). An Activity-Theoretical Method for Studying Dynamics of User-Participation in IS Design. S. Bjornestad, A. Morch, & A. Öpdahl (Eds.), *IRIS 24, 24th Information Systems Research Seminar in Scandinavia*, Ulvik in Hardanger, Norway.

Johnson, R.R. (1998). *User-centered technology : a rhetorical theory for computers and other mundane artifacts.* Albany: State University of New York Press.

Kling, R., & Public Policy Research Organization (University of California, I. (1977). *The organizational context of user-centered software designs*. Irvine: Dept. of Information and Computer Science and Public Policy Research Organization University of California.

Knorr, K.D. (1979). Tinkering toward success: Prelude to a theory of scientific practice. *Theory and Society*(8), 347-376.

Knorr-Cetina, K., Savigny, E.v., & Schatzki, T.R. (2001). *The practice turn in contemporary theory*. London ; New York: Routledge.

Latour, B. (1987). *Science in action : how to follow scientists and engineers through society*. Cambridge, Mass. :: Harvard University Press,.

Lave, J., & Chaiklin, S. (1993). *Understanding practice : perspectives on activity and context*. Cambridge ; New York, N.Y.: Cambridge University Press.

Leontjev, A.N. (1978). *Activity, consciousness, and personality*. Moskow: Progress.

Lessig, L. (1999). *Code and other laws of cyberspace*. New York :: Basic Books.

Levi-Strauss, C. (1966). *The savage mind*. [Chicago]: University of Chicago Press.

Lévy-Bruhl, L. (1978). *Primitive mentality*. New York: AMS Press.

Lévy-Bruhl, L. (1985). *How natives think*. Princeton, N.J.: Princeton University Press.

Mead, G.H., & Morris, C.W. (1974). *Mind, self, and society : from the standpoint of a social behaviorist*. Chicago: University of Chicago Press.

Miettinen, R. (1998). Object Construction and Networks in Research Work: The Case of Research on Cellulose Degrading Enzymes. *Social Studies of Science, 28*.

Norman, D.A. (1990). *The design of everyday things* (1st Doubleday/Currency ed.). New York: Doubleday.

Norris, K.S. (1993). *Dolphin days : the life & times of the spinner dolphin*. New York: Avon Books.

Nutch, F. (1996). Gadgets, Gizmos, and Instruments - Science for the Tinkering. *Science Technology & Human Values, V21*(N2), 214-228.

Patent Resources Group, & Bender, D.v. (1969). *Software protection by trade secret, contract [and] patent law; law, practice and forms*. Washington.

Perrow, C. (1999). *Normal accidents : living with high-risk technologies : with a new afterword and a postscript on the Y2K problem*. Princeton, N.J.: Princeton University Press.

Rossi-Landi, F. (1983). *Language as work & trade : a semiotic homology for linguistics & economics*. South Hadley, Mass.: Bergin & Garvey Publishers.

Samuelson, P., & University of California, B.E.T.O. (1997). Copyright in cyberspace.

Scribner, S., & Cole, M. (1981). *The psychology of literacy*. Cambridge, Mass.: Harvard University Press.

Smith, B.C. (1996). *On the origin of objects*. Cambridge, Mass.: MIT Press.

Turnbull, D. (2000). *Masons, tricksters and cartographers : comparative studies in the sociology of scientific and indigenous knowledge*. Amsterdam
Abingdon: Harwood Academic ;
Marston.

United States. Congress. Office of Technology Assessment (1987). *Intellectual property rights in an age of electronics and information*. Malabar, Fla.: R.E. Krieger.

United States. Congress. Office of Technology Assessment (1990). *Computer software & intellectual property*. Washington, D.C.: Congress of the U.S. Office of Technology Assessment : For sale by the Supt. of Docs. U.S. G.P.O.

United States. Congress. Office of Technology Assessment (1992). *Finding a balance computer software, intellectual property and the challenge of technological change : summary*. Washington, D.C.?: Congress of the U.S. Office of Technology Assessment.

Wartofsky, M.F. (1979). *Models. Representation and the Scientific Understanding*.
Dordrecht: D. Reidel.