A Presentation at OWLS: The Oxford Workshop on 'Libre Source'
Convened at the Oxford Internet Institute, 25-26th June 2004

Free & Open Source Software Developers and 'the Economy of Regard': Participation and Code-Signing in the Modules of the Linux Kernel

By

Jean-Michel Dalle
Jean-Michel.Dalle@upmc.fr

Paul A. David*
pad@stanford.edu

Rishab Aiyer Ghosh
rishab@dxm.org

& Frank A.Wolak
wolak@stanford.edu

First draft: 24 March 2004
This version: 17 June 2004

ABSTRACT

We attempt to illuminate the interplay of decentralized, micro-level decisions that shape the allocation of individual voluntary software development efforts on the part of collectivities (communities) of agents, distributing their work among the distinct 'modules' (or 'packages' of code) that form large and complex "open source" system products. The paper integrates several distinct sources of information and methods of analysis, in particular: (a) behavioral generalizations of a sociological character deriving from expert-participant observation; (b)descriptive quantitative data extracted from open source code, exposing the technical features (e.g., size, technical dependency structure) of the modules forming the Linux kernel and (c) distributions of authorship credits among the modules in the same releases of the Linux kernel. This material is used to draw statistical inferences about factors affecting individual developers' decisions affecting the distribution of code-writing efforts within this large, emblematic project, based upon estimation of the equations of a econometric model of code-signing and participation behaviors. The approach pursued is a first step towards more complex quantitative analyses that will exploit the self-revealing ontological features of open source software, and the array of new tools for machine extraction of data from code repositories. A concluding discussion examines the implications of the findings for the construction of agent-based stochastic simulation models designed to reveal properties of self-organized community-mode software production.

Acknowledgements

The research reported in this presentation could not have been undertaken without the financial support received by the Stanford Institute for Economic Policy Research (SIEPR) Project on the Economics of Free and Open Source Software in the form of grant awards by the National Science Foundation program on Digital Technologiy and Society: IIS-0112962(2001-04) and IIS-0329259(2003-05). [See: http://siepr.stanford.edu/programs/OpenSoftware_David/OS_Project_Funded_Announcmt.htm] David and Wolak are grateful also for the support provided by SIEPR's Director and staff, and the Stanford University Vice-President for Undergraduate Education's program for Undergraduate Research Experience.

**Contact author:* P. A. David, Oxford Internet Institute, 1 St. Giles', Oxford OX1 3SJ, U.K.
Tel. +44+(0)1865 287210; email: pdavid@herald.ox.ac.uk

Free & Open Source Software Developers and 'the Economy of Regard': Participation and Code-Signing in the Modules of the Linux Kernel

0. Introduction

We report here on recent results from one phase of a collaborative program of analytical and empirical research on the economic organization, performance and viability of the free/libre and open source (F/LOSS) mode of software production. The focus of this inquiry concerns the interplay of decentralized, micro-level decisions that shape the allocation of individual voluntary software development effort on the part of individual agents, distributing these among the distinct 'modules' (or 'packages' of code) that form large and complex "open source" system products.

The paper integrates several distinct sources of information and methods of analysis, in particular: (a) behavioral generalizations of a sociological character deriving from expert-participant observation; (b)descriptive quantitative data extracted from open source code, exposing the technical features (e.g., size, technical dependency structure) of the modules forming the Linux kernel and (c) distributions of authorship credits among the modules in the same releases of the Linux kernel. This material is used to draw statistical inferences about factors affecting individual developers' decisions affecting the distribution of code-writing efforts within this large, emblematic project, based upon estimation of the equations of a econometric model of code-signing and participation behaviors. The approach pursued is a first step towards more complex quantitative analyses that will exploit the self-revealing ontological features of open source software, and the array of new tools for machine extraction of data from code repositories.

A concluding discussion examines the implications of the findings for the construction of agent-based stochastic simulation models designed to reveal properties of self-organized community-mode software production.[1] The development and parameterization of this model, itself part of a larger simulation structure, can thus be informed by available micro-level evidence relating to the behaviors of F/LOSS developers and the technical properties of modularly designed code. The dual methodological purpose served by this approach is to integrate key empirical findings, and expose some of the links between characteristic micro-level features of "the open source way of working" and observed meso-level performance properties of the F/LOSS development process.

The material to be presented here organizes itself naturally under three main headings. Section 1 briefly considers the question of individual motivations and its bearing upon the decentralized organization and spontaneous coordination that are characteristic features of the open source software production mode – and other distributed multi-agent processes of "collective invention." Attention is directed to the set of "motivations-at-the-margin" that are thought to influence contributing developers' decisions about the specific problems or tasks that they will tackle in the course of participating in a particular software project. Building upon the insightful commentary of an informed observer, Eric Raymond (1999), we formulate a number of behavioral propositions that give concrete form to the hypothesis that choices of this kind are systematically affected by shared perceptions of their likely

[1] See Dalle, David and Steinmueller (2002), for further description of the integrated research program that is being pursued by the Stanford Project working in collaboration with colleagues at **GSyC** – Informatics (Universidad Rey Juan Carlos), **IMRI** (University of Paris-Dauphine), **MERIT** (University of Maastricht), and **SPRU** (University of Sussex).

consequences in terms of "peer regard" (recognition and esteem) within the open source developers' community.

In Section 2 sets out the methodology for generating indirect empirical evidence with which to confront the view of F/LOSS developers as responsive to the socially defined norms and incentives of a particular instantiation of "the economy of regard" – a resource allocating structure of human interaction that has been described by Avner Offer (1997) as situated between reciprocated gift exchange and the market. An econometric model for the joint micro-level decisions about participation in the development of a sub-project (module) and signing the code that committed to that sub-project, is presented. Inasmuch as individual participation can only be observed (from the code base itself) when contributed lines of code are signed ("credited") it is necessary to take the determinants of the latter behavior into account when studying participation, and there is reason to hypothesize also that the extent of actual participation in a sub-project may influence code signing behaviors among the participants. The statistical results obtained by estimating this model on the basis of data extracted from the code of Linux kernel Version 2.5.25 represent both a conceptual and an econometric improvement upon the preliminary analysis described by David and Ghosh (2003).

The paper concludes in Section 3 with a discussion of the interpretations that can be placed upon the econometric results, and the implications these have for debates about the salience of considerations of "peer regard" and "collegiate reputation" effects among the incentives to which developers of open source software systematically respond. In addition to underscoring the importance of the operational distinction between self-reported, awoved motivations for participation in free and open source projects and behaviorally reveal micro-motives – or 'motivations at the margin' – that govern the specific forms and extent of participation with a given project, our analysis points to the significance of value norms guiding developers' dynamic responses to the evolving project-environment that is being formed by their collective efforts and interactions. Thus, we briefly consider whether micro-level actions that are hypothesized to be motivated-at-the-margin by the quest for "peer regard" may be a source of the highly skewed distributions of F/LOSS project attributes, which exhibit extreme concentration in the inter-module distribution numbers of developers contributing to, and the code size attained by distinct modules within large projects such as the Linux kernel.

1. Human Motives, Incentives and Non-market Allocation Mechanisms

Eric Raymond (1999) and other observers of open source communities suggest that choices made by developers regarding where and how to contribute their expertise and effort are guided by perceptions of their likely consequences in terms of recognition and esteem (including self-esteem) from those working on the projects in question, and from peers in the open source community at large. Economists have suggested that career considerations, involving expectations of future material benefits from "signaling" expertise through open source contributions, are powerful in motivating the voluntary participation of developers.[2] The latter interpretation renders the quest for the approbation of peers in a community of

[2] See Lerner and Tirole (2002). Dasgupta and David (1994), in discussing the competition for talent between "open science" and "proprietary R&D", proposed "signalling" benefits as a rationale for the readiness with which young scientists (even those who expect eventually to have an industrial research careers) initially compete for lower-paid, non-tenure track academic appointments as postdoctoral fellows and lecturers. One attraction of academic appointments--especially for those who regard their line of research to be particularly promising—lies in the greater freedom afforded to those who hope to gain attention by publishing their findings.

practice as "instrument," that is to say, sought to accomplish other ends rather than for its inherent value, or for intrinsic qualities of the quest itself. Variants on the instrumental motivations theme have alternatively stressed the immediate benefits some user-developers derive through the enhanced functionality of the programs to which they contribute (see, e.g., Lakhani and von Hippel (2003), or the acquistion of software skills and corresponding improvements in their future opportunities for gainful employment in the industry.[3]

These intriguing and plausible interpretative hypotheses about the force of various classes of motives current rest largely on theoretical speculation, on analogies with the sociological studies of academic "open science" communities, or on ethnographic material gathered by participant observers of F/LOSS communities at work. Why individuals contribute their time and skill to software development projects whose results are made available for general use under free and open source licenses is a question that has intrigued many economists. Such behavior appears anomalous in the context of mainstream economic analysis, which clings to the simplifying supposition that work of any sort must entail "disutility;" and that individuals who sacrifice the satisfactions of'leisure' compensating payment that (whether directly or indirectly) takes the form of an enlarged command over goods and services that are the natural sources of "utility." But, to appreciate the attention that economists have devoted to trying to identify the 'why' of F/LOSS participation, it is not necessary for us to argue the merits and deficiencies of that particular analytical framework. One hardly could hardly claim to understand the mobilization and allocation of the human resources engaged in such projects without knowing something about the motivations of the individuals involved. Yet, it is important to notice the difference between the very general issues of motivation that appear to concern many recent contributions to economics of open source software, and the more restricted scope of the present discussion.

A rather different, less individualistic framework for understanding recurring voluntary transactions that are not highly personalized and in which pecuniary compensation plays little if any significant part is proposed by Avner Offer (1997): his conceptualization of "the economy of regard" is a system of reciprocated exchanges that is situated "between the gift and the market." This is helpful in the present context for two principle reasons. Firstly, it moves the discussion away from the essentially atomistic mode of analysis familiar cartel theory and public finance economics, where agents are depicted as passively deciding whether to free-ride or join a pre-existing collective organization whose existence they view to be independent of their personal actions. Instead, it invited analysis of the instrinsic satisfactions that people may derive from interactive, community-creating aspects of participation – particularly in middling and large F/LOSS projects.

Secondly, the notion of the 'economy of regard' being distinct from the classical "gift economy," and positioned the intermediate space of non-market social systems based upon indirectly reciprocated, and partially personalized exchanges, deftly characterizes the larger F/LOSS projects as belonging to a broader array of epistemic communities institutionalized communities of practice. Although some commentators have sought to"naturalize" participation in F/LOSS development by presenting such activities as simply another instance of gift exchange,[4] the"gift model" derived from Mauss (1925) has features that are not

[3] See Waterman (May 2003) for a careful review of these and other motivational explanations, as well an attempt to formalize these hypotheses in a model the defines the boundaries of the multi-dimensional space within which F/LOSS production is likely to be sustainable.

[4] Raymond (1999: pp. 80-81), focusing upon the absence of conditions of scarcity that deprives exchange cultures of their ability to command effort and define status, write: "…examined in this way, it is quite clear that the hacker society is *a gift culture*." For critical discussion and rejection of this interpretative framework, see

completely congruent with the transactions among developers of open source software. True, there are some strong elements of resemblance, insofar as the "gift" of code-writing, or bug-patching is a voluntary transfer; and there is some expectation of reciprocity (more indirect than direct) fulfillment of which is a matter left to for individual discretion as to the timing and magnitude of the reciprocal offering.[5] But, thereafter the analogy becomes progressively more strained: in gift systems esteem for others is what the proferred gift is meant to communicate, and personalized (of ceremonially prescribed) gifts attest to the giver's "regard." Such clearly is not the case in distributed, open source software communities, where interactions may be quite anonymous and impersonal, yet nonetheless valued. Nor can one see that the putative "gift" made in the form of the F/LOSS developer's "commit" serves to establish a repetitive, self-enforcing bond, facilitating subsequent reciprocated transactions.[6]

This paper, by contrast, presents objective, quantitative evidence to substantiate the view that F/LOSS developers behavior is shaped by an "economy of peer regard." Raymond's propositions are in some important respects much more specific than the general notion that "reputation effects" deriving from peer evaluation of one's public contributions motivate the voluntary participation of people in developing open source software. In pointing to a hierarchy of ordinal 'valuations' based upon objective attributes of the contributions that individuals might make to the collective enterprise, Raymond's work suggests an approach to empirical enquiry: if "community regard" differentiates among technical contributions, then it should be possible to find evidence that "regard-seeking" individual actions are more likely to be elicited by some opportunities to make programming contributions to a software project than by others. This is the approach we explore in Section 2 of this paper. Our evidence about the behavioral responses of open source developers to considerations of that kind also does not permit us to distinguish among the variety of reasons why the recognition and esteem of others is a matter of concern, and hence a motor for action. Whether "peer regard" is sought for its possible instrumental value (as a source of income, career advancement), rather than for its intrinsic satisfactions, is not an issue which can be resolved with the materials under analysis here. But, as will be seen, our purposes do not require pinpointing that the

---

e.g., Ghosh (1998), Kuwabara (2000). Similar, anthropologically inspired interpretations of transactions among members of open science communities as "gift exchanges", were similarly found inadequate by Dasgupta and David (1987).

[5] Reciprocation in the case of F/LOSS development remains unenforced, save by moral compulsion, which is also the situation that Mauss described among the Polynesian people, where the "spirit of the gift" weighs heavily upon the recepients until they can relieve oneself of the burden by presenting a gift to some other person.. Gambardella and Hall (2004) suggest that the GNU GPL thereby serves as a coordination point that increases the likelihood of the community attaining what the game theory literature characterises as an "altruistic equilibrium."

[6] Unlike the "spirit of the gift" in Mauss's account (see note 5, above), the restrictions of the "copyleft" principle embedded in the licensing terms of GNU GPL software cannot not compel reciprocation from the receivers of open source software; it does impose a form of reciprocation upon the compliant who choose to modify and distribute the code they have received from the F/LOSS producing collectivity. Gambardella and Hall (2004) suggest that the GNU GPL thereby serves as a coordination point that increases the likelihood of the community attaining what the game theory literature characterises as an "altruistic equilibrium." On OSS licensing issues, see the lucid exposition in McGowan (2003).

underlying goals and aspirations of individual actors – as distinct from their more proximate objectives.

1.1. Motivations-at-the-margin and decentralized creative resource allocation

In research on this subject it hardly would be possible to entirely eschew taking account of what has been discovered about the variety prospective rewards – both material and psychic – that may be motivating individuals to write free and open source software. It is only reasonable to suppose that these will in some way influence how developers allocate their personal efforts in this sphere. At this stage it is not necessary to enter fully into the question "Why do they do it?" – the much discussed and debated motivational preoccupation of many studies of "the open source phenomenon."[7] Economists, more than other social scientists, have been overly pre-occupied by the seemingly anomalous observation that a large number of people around the world have been contributing their software programming and related skills to code-creating enterprises from which they have no expectations of deriving any direct pecuniary rewards. But this putative "anomaly," like all scientific anomalies, arises in relation to some particular strongly held a priori expectations --"a theory," in other words, about the way the world works. Mainstream economic theory about the sources of human motivation, resting as it does upon the remnants of Benthamite speculations, is so lacking in psychological sophistication that virtually all individual behavioral phenomena could be, and were categorized as "anomalous."[8] Conventional economic analysis is far more usefully engaged where, instead of providing an answer to the question "Why is this done?" the subject of the conversation is changed to "In what circumstances is this done?" and "When is rather more (rather than less) of this done?"[9] This finesse, substituting analysis of what might be referred to as "motivation at the margin," makes better use of the insights that the economist's *métier* can provide about the way specific incentives and constraint affect the incremental allocation of resources.

The difference in perspectives is mirror by the differences in the character of the responses elicited from open source developers in response to two different kinds of enquiries into the factors motivating their actions: questions on reasons for engaging FOSS invite consideration of broad, socially-oriented rationales, whereas individualistic motives are more likely to be surfaced by asking what the person chose one rather than another form of

---

[7] For a discussion of the literature and results of a large survey of F/LOSS developers' characteristics and attitudes bearing on motivation, see, e.g. Ghosh (2003), Ghosh, Glott and Robles (2003). These findings from the 2002 FLOSS survey of developers indicate that "ideological" rather than "instrumental" motives are frequently identified as salient when developers are ask about their reasons for participating in open source software development. The 2003 FLOSS-US survey, responses to which were significantly less dominated by residents of western Europe, revealed the same thing: 78-79 percent of respondents gave very important or important weight to both of the following reasons: "we should all be free to modify the software we use" and "as a user of free and open source software, I wanted to give back something to the community."

[8] Indeed, the new and growing sub-discipline of "behavioral economics" might be seen as a belatedly constructive response to the challenge posed by this embarrassment of anomalies.

[9] See, e.g., the analysis by Waterman (2003), which responds to the question "Why?" by seeking to identify the "economically viable niches" in which free and open software development activity is likely to be sustained. This approach can be seen to belong to a generic style of economic research that resembles statistical mechanics (see Sanderson 1974). Rather than trying to predict the micro-level behaviour of the agents, the explanatory strategy seeks to account for robust empirical regularities, involving population ensembles, by examining the constraints on agents actions that determine the boundaries on distributions of observable "events" or "outcomes" (resource use patterns).

participation. Moreover, the desire of individuals for social approbation, or the affirmation of identification with a social group or community, is more likely to elicit responses that echo widespread expressions of community purpose when questions of motivation invite socially-oriented rationales. Thus, it is not surprising that the preliminary analysis of data from the FLOSS-US 2003 Survey by David, Waterman and Arora (2004) finds a marked contrast between the tenor of responses to the question about reasons to participating in FOSS development—questions of the same character as those posed by the FLOSS 2002 Survey, and a follow-on set of questions that asked why the respondent had chosen to work on their first project. The latter responses are significantly focused on instrumental considerations, notably, the individual's interest in the technical characteristics of the project either for skill development or for personal use.

But the supposition that FOSS developers can be profiled by reference to a stable set of motivations is itself questionable; there is are theoretical and empirical grounds for supposing that the reasons people offer for their actions do not remain unchanged. Although economists in mainstream tradition have resisted evidence that tastes and preferences are endogenous, they appear only grudingly to accept the possibilities that learning and habituation may result in preferences that are altered by experience of interactions with the environment. Cognitive dissonance theory suggests, in addition, that self-avowed motives may well change under essentially internal psychological pressures; people find "reason" ex post for behaviors to which the individual may be committed by unconscious or external constraints. Does this help explain how "motives for doing FOSS appear to evolve with experience" in the FLOSS survey? Evidence reported by Rudiger Glott (2004), based upon an analysis of the FLOSS-2002 Survey responses, indicates that for more than half the respondents whose initial reasons for participating in FOSS development were so diffuse as to resist categorization, their stated reasons for continuing to participate indicate a bi-polar movement towards either more sharply delineated "ideological" commitment to the community, or a dominant "instrumental" rationale. In the latter category, 'improving software skills" – already present among the focused motives for beginning participation, emerges saliently among the reasons for sustained participation.

1.2 Developer's motives and production of code in "C-mode" vs "I-mode"

Although it is not necessary at this juncture to enumerate the array of diverse "devices and desires" – that is to say, the gamut of motives from the pragmatically instrumental to the purely felicific impulses– that may impel boys and grown men (for there are precious few girls and women) to spend their more rather than less time in their evenings, weekends and vacations writing code and fixing "bugs" in software, it is important for what follows that we put to one side one widely recognized, and undeniably important class of reasons why open source code is produced. These could be described as the individual incentives that give rise "independent user-implemented innovations."[10] Indeed, this term may well apply to the great mass of identifiably discrete open source software projects, because a major consideration

[10] The term evidently derives from von Hippel's (2001, 2002) emphasis on the respects in which open source software exemplifies the larger phenomenon of "user-innovations. Of the 1562 respondents to the FLOSS-US Survey (see David, Waterman and Arora (2003)), 56 percent scored as a "very important" or "important" the following reason for their participation in free and open software development: "I needed to perform tasks that could only be done with modified versions of existing software"; 53 percent gave same important to: "I needed to fix bugs in software that I was using." The great mass of open source projects that appear on sites such as SourceForge and FreshMeat are small, and typically involve only a few identified developers at most. Whether most of thse cases fit von Hippel's category of "user-innovators" cannot be determined from their size alone.

driving many individuals who engage in the production of open source would appear to be the direct utility or satisfaction they expect to derive by using their creative outputs. The power of this motivating force obviously derives from the property of immediate efficacy, which has been noticed as a distinctive feature of computer programs. But, no less obviously, this class of motives will be most potent where the utilitarian objective does not require developing a large and complex body of code, and so can be achieved quite readily by the exertion of the individual programmer's independent efforts, or those of a small number of collaborators.

"Independent" is the operative word here, for it is unlikely that someone writing an obscure driver for a newly-marketed printer that he wishes to use will be at all concerned about the value that would be attached to this achievement by "the F/LOSS community." The individuals engaging in this sort of software development may regard themselves as belonging in every way to the free software and open source movements, and may be committed to using open source development tools. Nevertheless, with respect to any particular project they have the real option of waiting until their project is substantially, or wholly completed before publicly revealing the source code, and thereby sacrifice their ability to exercise their legal rights to exploit their creation for commercial gain the under protections afforded by the copyright statutes. Being essentially isolated from significant collaboration in development process, the issue of formal disposition of the authorship rights in a collective creation will not arise of necessity in organization the production process, and so the entire issue of the licensing terms may be deferred until the code is written. This option, however, typically will not be available for projects that contemplate enlisting the voluntary contributions of numerous developers who are not co-located. In such cases, the intention to mobilize the resources available in the free and open source software development communities follows from the technical scope of the undertaking, and provides compelling reasons to announce a licensing policy *ex ante*.[11]

For all intents and purposes software production activity in such circumstances stands apart from the efforts that entail participation in collective developmental process, involving successive releases of code and the cumulative formation of a more complex, multi-function system. We will refer to the latter as F/LOSS production in "community-mode" or, for convenience *C-mode,* contrasting it with software production in *I-mode*. Since I-mode products and producers, almost by definition, tend to remain restricted in their individual scope and do not provide as direct an experience of social participation, the empirical bases for generalizations about them is still very thin, too thin to provide interesting behavioral propositions that could be incorporated in a formal model. Besides, significant issues of resource mobilization and coordination in production, almost by definition, do not present themselves in the case of software generated in *I-mode*.[12] Consequently, our attention here

[11] The present aspect of our investigations abstracts from the possible effects upon the organization of F/LOSS production that may arise from choices among the licensing options, e.g., adding the new code the corpus of copyright-protected "libre" software that is released under the GNU GPL and LGPL licenses, rather than using the BDS or other variant licenses, or, indeed, simply putting it into the public domain. Such analytical attention as the question of open source software licensing has received from economists typically ignores the possibility of that consideration of such (production) effects might shape the licensing decision. See, e.g., Lerner and Tirole (2002), Gaudeul (2003). In a recent and interesting departure from the "mainstream" tradition, however, Gambardella and Hall (2004) examine a simple model that allows for the connection between the choice of licensing arrangements and the optimality of the production regime for information goods.

[12] This is not to say that there are no questions of interest for economic analysis. As we have pointed out, the decision to release code under open source licenses, rather than keeping it closed and licensing its commercial use on terms available under the copyright statutes, or even patenting a process implemented by an embedded form of the software, are options among which developers make choices. In a sense these choices parallel those

focuses exclusively upon creating a suitable model to simulate the actions and outcomes of populations of F/LOSS agents that are working in *C-mode*.

It would be a mistake, however, to completely conflate the issue of the sources of motivation for human behavior with the separable question of how individuals' awareness of community sentiment, and their receptivity to signals transmitted in social interactions, serves to guide and even constrain their private and public actions; indeed, even to modify their manifest goals. Our stylized representation of the production decisions made by F/LOSS developers' therefore does not presuppose that career considerations of "ability signalling," "reputation-building," and the expectations of various material rewards attached thereto, are dominant or even a sufficient *motivations* for individuals who participate in *C-mode* projects. Instead, it embraces the weaker hypothesis that awareness of peer-group norms significantly influences (without completely determining) micro-level choices about the individuals' allocation of their code-writing inputs, whatever assortment of considerations may be motivating their willingness to contribute those efforts.[13]

1.3 Behavioral foundations for C-mode production of software

An important point of departure for our work is provided by a penetrating discussion of the operative norms of knowledge production within *communities* developing free/libre and open source software (F/LOSS), which appears in Eric Raymond's essay, "Homesteading the Noosphere" (Raymond, 1999: pp. 65-111). Within the "noosphere" – the "space" of ideas, according to Raymond -- software developers allocate their efforts according to the relative intensity of the reputation rewards that the community attaches to different code-writing "tasks." The core of Raymond's insights is a variant of the collegiate reputational reward system articulated by sociological studies of open science communities: the greater the significance that peers would attach to the project, to the agent's role, and the greater is the extent or technical criticality of his or her contribution, the greater is the "reward" that can be anticipated. Although Raymond is an astute participant-observer of these F/LOSS communities, and his sociological generalizations have the virtue of inherent plausibility, these propositions remain to be validated by independent empirical tests.[14]

Caricaturing Raymond's more nuanced discussion, we stipulate that (a)launching a new project is usually more rewarding than contributing to an existing one, especially when several contributions have already been made; (b) early releases typically are more rewarding

---

facing an academic researcher who have obtained a result that is readily publishable, and must decide whether to disclose it or first explore the possibilities of commercial exploitation, either by starting a enterprise to "work" the knowledge as a trade secret, or to devise an application that would secure a patent and the prospect of licensing income. In the simple case in which the discovery can be viewed as an exogenous event, and the discover independent – in the sense that she can legally hold the exploitation rights, this is the classic IPR decision issue; it poses no significant new issues in the case of software.

[13] It will be seen that the probabilistic allocation "rules" derive from a set of distinct community "norms," and it will be quite straightforward within the structure of the model to allow for heterogeneity in the responsiveness to peer-influence in this respect, by providing for inter-individual differences in weighting within the rule-set. This may be done either probabilistically, or by creating a variety of distinct "types" of agents and specifying their relative frequencies in the population from which "contributions" are drawn. For the purposes of the basic model presented here, we have made a bold simplification by specifying that all potential contributors respond uniformly to a common set of allocation rules.

[14] See e.g. Lakhani & Wolf (2003), Oh & Hans (2003), Niedner et al. (2003), and the systematic survey or interviews with representative samples of F/LOSS community participants done by the FLOSS survey (Ghosh et al., 2002) and its US counterpart – "FLOSS-US" – at Stanford University.

than later versions of project code; (c) there are some rewarding projects within large software system that are systematically accorded more "importance" than others. One way to express this is to say that there is a hierarchy "peer regard," or reputational significance, attached to the constituents elements of a family of projects, such that contributing to the Linux Kernel is deemed a (potentially) more rewarding activity than providing Linux implementation of an existing and widely used applications program, and the latter dominates writing an obscure driver for a newly-marketed printer. To this list we would append another hypothesized "rule": (d) within each discrete project, analogously, there is hierarchy of peer-regard that corresponds with (and possibly reflects) differences in the structure of meso-level *technical* dependences among the "modules" or integral "packages" that constitute that project. In other words, we postulate that there is lexicographic ordering of rewards based upon a discrete, technically-based "tree-like" structure formed by the successive addition of project components. Lastly, for present purposes is can be assumed that (e) new projects are created in relation to existing ones, so that always is possible to add a new module in relation to an existing one, to which it adds a new functionality. The contribution made by initiating this new module (being located one level higher in the tree) will be accorded less significance than its counterparts on the structure's lower branches.

Thus, our model postulates that the effort-allocation decisions of agent's working in *C-mode* are influenced (*inter alia*) by their perceptions concerning the positioning of the project's packages in a hierarchy of peer-regard; and, further, stipulates that the latter hierarchy is related to the structure of the technical interdependences among the modules.

For present purposes it is not really necessary to specify whether dependent or supporting relationships weigh receive the relatively greater weight in this "calculus of regard." Still, we will proceed on the supposition that modules that are more intensely implicated by links with other packages that include "supportive" connections reasonably are regarded as "germinal" or "stem" sub-routines[15] and therefore may be depicted as occupying positions towards the base of the tree-like architecture of the software project. Assuming that files contributed to the code of the more generic among the modules, such as the kernel or the memory manager of an operating system (e.g., Linux) would be called relatively more frequently by other modules, this might accord them greater "criticality"; or it might convey greater notice to the individual contributor that that which would apply in the case of contributions made to modules having more specialized functions, and whose files were "called" by relatively few other packages.

For the present purposes, Raymond's rules be restated as holding that: (1) there is more "peer regard" to be gained by a contribution made to a new package than by the improvement of existing packages; (2) in any given package, early and radically innovative contributions are more rewarded than later and incremental ones; (3) the lower level and the more generic a package, the more easily a contribution will be noticed, and therefore the more attractive a target it will be for developers. Inasmuch as "contributions" also are acknowledged by Raymond as correcting "bugs of omission", each such contribution – or "fix" – is a patch for a "bug", be it a simple bug, an improvement, or even a seminal

[15] Cautiousness is needed when using the word "root" to designate the germinal modules, because importing that term from from the arboral metaphor may be confusing for programmers: we are told by one informant that in "Unix-speak" the system administrator is called "root", and the top of the file structure, likewise, is "root." Indeed, our hypothesized "dependency tree" might also be in some extent related to the more familiar directory tree structure, but this correlation is likely to very imperfect.

contribution to a new package. Therefore every contribution is associated with a variable expected payoff that depends on its nature and "location"[16].

The decision-problem for developers is then to choose which "bug" or "problem" will occupy their attention during any finite work interval. We find here another instance of the classic "problem of problem choice" in science, which the philosopher Charles S. Pierce (1879) was the first to formalise as a microeconomic decision problem. But we need not go back to the static utility calculus of Pierce. Instead, we can draw upon the graph-theoretic model that more has recently been suggested by Caracole and Dale's (2000) analysis of the way that the successive choices of research agendas by individual scientists can aggregate into collective dynamic patterns of knowledge accumulation. The latter modelling approach is a quite suitable point of departure, precisely because of the resemblance between the reputation game that Raymond (1999) suggests is played by open-source software developers and behavior of open science researchers in response to collegiate reputational reward systems, as described by Dasgupta and David (1994). Although we treat agents' "problem-choices" as being made independently in a decentralised process, they are nonetheless influenced by the context that has been formed by the previous effort-allocating decision of the ensemble of researchers. That context can be represented as the state of the knowledge structure accumulated, in a geological manner, by the "deposition" of past research efforts among a variety of "sites" in the evolving research space – 'the noosphere' of Raymond's metaphor of a "settlement" or "homesteading" process.

Having indicated the goal towards which this paper is advancing, a pause is order before continuing with the presentation in Section 3 of an agent-based simulation model. The latter puts these behavioral foundations we have been examining to work, by specifying rules guiding the micro-level decisions of individual agents in s "virtual" open source community project. It is therefore pertinent to ask whether any empirical evidence can be adduced in support of the central supposition, which is that open source developers are embedded in "an economy of peer regard" that resembles a market economy is providing signals and incentives that harness their motives and thereby are able to guide their individual and collective efforts. The material in the next section is an interim answer to that question.

2. Open-Source Code-Signing and Developers' Participation Decisions

This section presents results obtained by estimating an econometric model of developers' decisions in regard to the signing of contributed source code, considering this jointly with the decision to participate in the development of one or another open source software project. Implementation of this approach to securing indirect evidence regarding developers' motivations-a-the-margin has not been attempted for the more complex situations raised by choices among different and unrelated projects, which may have "local" norms, *mores* and governance procedures, including differences in policy with regard to whether or not "commits" carry the email signature of the contributors.

INSERT:

[16] Note that here we neglect, for the moment, the possibility that bugs can become more attractive "targets" because they've existed for long and have thus drawn the attention of the community of developers, and also more specific peer assessments of the "quality" of patches.

Code-signing is a matter of individual choice, but it is choice conditioned by social and organizational context. Developer survey (FLOSS) reports responses. Give figures.
But it is unfortunate that these responses are not situated with reference to projects in which the individuals have worked. There are two possibilities to be considered: code-signing predilections can be learned with community experience, if there are community norms regarding the practice.

Code is signed in the Linux kernel, although not in other large projects, such as Apache. Alternatively, if we suppose preference are invariant and strong for signing code, this could affect choices of participation among communities.

These are interesting issues that nonetheless lie beyond the bounds of the present study, for we confine ourselves to looking at behaviors within those who participate in a single project, for which there is a permissive policy regarding code signing. It is therefore possible that those who regard it as important to sign, and those who do not are both in the population under examination. How they distribute themselves is the question.

The identity of committers may be denoted in annotations that are found in the evolving CVS code base. In this study, however, we utilize data extracted from successive releases of source code using the CODD Dependency Analyzer algorithm developed by Robles and Ghosh (2004). For this exercise we work with data extracted from the code of the 169 distinct "code packages" or modules that can be identified as the constitutive elements of Version 2.5.25 of the Linux kernel.[17]

The model and estimation procedure described here improves upon the preliminary work reported by David and Ghosh (2003) in two principal respects: first, it utilizes all the available information on code-signing, rather than by excluding those modules of the kernel in which all of the code was "credited" (i.e., signed by the committers); second, the estimates take account of the fact that the total number of developers contributing to a module (or 'package') is observed only when all the code in the module is signed. As a result, we have substantially new, and in some respects different findings to report. But, happily, these admit of interpretations that are consistent with according a motivational role to developers' considerations of "reputational rewards" – including the physic enjoyment of collegial esteem ("peer approval").

This analysis uses information on a cross-section of the 169 modules identified as distinct elements within the code of Linux kernel Version 2.5.25 [A total of 180 code packages were identified in this and two earlier releases, Version 1.0 and Version 2.0.3, but twelve of those directories were no longer present in Version 2.5.25.]. The data compiled for each package are as follows:

numbytes = the total number of bytes of code written,
uncredit = the total number of bytes of code uncredited to any developer,

[17] See Ghosh (2003) on the methodology of data extraction and its application to Versions 1.0, 2.0.3 and 2.5.25 of the Link kernel; Ghosh and David (2003) for descriptive findings on the nature and structure of the dynamics of authorship and technical topologies in the this project.

ndevelop = the total number of signing developers on these bytes of code,
supnum = the number of modules that this package supports (i.e., that "call it"),
depnum = the number of modules that the package depends upon (i.e., that "it calls").

2.1 Specifications of the estimation model

There are three equations in this model. The dependent variable modeled in the first equation is the logarithm of the ratio uncredited- to credited-bytes for modules that have a non-zero amount of credited code (i.e., all the packages). This variable is assumed to depend on the total number of developers working on a project, which is only observed when there are no uncredited bytes on a project. The second equation predicts the total number of developers on a package, as distinct from the total number of developers credited as (signed) contributors to that package. The third equation summarizes the developer's decision of whether or not to leave all of the code committed unsigned, and hence "uncredited." The model assumes all of these decisions involve common unobservables in each of the three equations.

We define the following three dependent variables

y_{1t} = log(uncredit/(numbytes - uncredit)) = logarithm of ratio of uncredited to credited bytes in the package assuming both uncredited and credited bytes are positive

y_{2t} = logarithm of total number of developers that worked on package (only those cases that signed = 1 is the value of totdev observed and equal to log(ndevelop).

y_{3t} = dummy variable that equals 1 if all of the bytes in package t are credited (belong to physical lines of codes that were signed).

Associated with each dependent variable is a set of regressors X_{1t}, X_{2t}, and X_{3t}, respectively, We posit the following three structural equations:

$$y_{1t} = X_{1t}\beta_1 + \gamma y_{2t} + \varepsilon_{1t} \quad (1)$$

$$y_{2t} = X_{2t}\beta_2 + \varepsilon_{2t} \quad (2)$$

$$y_{3t} = X_{3t}\beta_3 + \varepsilon_{3t} \quad (3)$$

where

$y_{3t} = 1$ if $y_{3t}^* > 0$, and y_{2t} is observed and y_{1t} is not observed;

$y_{3t} = 0$ if $y_{3t}^* \le 0$ and y_{2t} is only known to exceed y_{2t}^{sign} = log(ndevelop), and y_{1t} is observed.

We assume that $\varepsilon_t = (\varepsilon_{1t}, \varepsilon_{2t}, \varepsilon_{3t})'$ is a mean zero normally distributed random vector with covariance matrix Σ,

$$\Sigma = \begin{bmatrix} \omega_{11} & \omega_{12} & \omega_{13} \\ \omega_{12} & \omega_{22} & \omega_{23} \\ \omega_{13} & \omega_{23} & 1 \end{bmatrix} \quad (4)$$

Define $ = ($1,$2,$3).

The log-likelihood for this model can be written as:

$$L(\beta,\alpha,\Omega) = \sum_{t=1}^{T} y_{3t} \ln\left\{ \int_{-\infty}^{\infty} \int_{-\infty}^{X_{3t}'\beta_c} \phi(z,(y_{2t} - X_{2t}'\beta_2), x \mid \Omega) dx dz \right\}$$
$$+ (1 - y_{3t}) \ln\left\{ \int_{y_{3t}^{sign}}^{\infty} \int_{X_{3t}'\beta_3}^{\infty_c} \phi((y_{1t} - (X_{1t}'\beta_2 + (X_{2t}'\beta_2)\alpha), (z - X_{2t}'\beta_2), x \mid \Omega^*) dx dz \right\} \quad (5)$$

where N(x,y,z | S) is density of a multivariate N(0, S) random variable, and S* is the covariance matrix of the multivariate normal random variable ,* = (,1t + ",2t, ,2t, ,3t).

2.2 Estimation results

Maximum likelihood estimates of this model are presented in the table below for the following definitions of X_{1t}, X_{2t}, and X_{3t}.

Table 2.1

Maximum Likelihood Estimates for the Model of Code-signing in Linux kernel 2.5.25

Variable	Parameter Estimate	Standard Error	t-statistic
Equation 3			
Constant	6.75369	0.96159	7.02346
Log(numbytes)	-0.549867	0.079588	-6.90894
Equation 2			
Constant	-5.51543	1.2634	-4.36556
Supnum	-0.00214	0.021799	-0.098155
Depnum	0.017049	9.19E-03	1.85577
supnum*depnum	1.86E-04	3.78E-04	0.493514
Log(numbytes)	0.614822	0.109806	5.59915
Equation 1			
Constant	-2.0408	1.62128	-1.25876
Supnum	0.011551	4.27E-03	2.7026
Depnum	-0.026702	0.018319	-1.45763
Log(total_developers)	-0.029421	0.277025	-0.106202
S11*	1.70537	0.191874	8.88797
S22*	0.564442	0.086245	6.54463
corr(,1t*,,2t*)	0.335085	0.309646	1.08215
corr(,1t*,,3t*)	-0.432752	0.432335	-1.00096
corr(,2t*,,3t*)	-0.6087	0.034821	-17.4811

3. Interpretations of the Findings

3.1 Reading the Statistical Estimates

(1) From the estimates for **Equation 3** it is seen that the probability that all the code contributed to a model will have been left unsigned (uncredited) varies inversely with the size of the package (measured in bytes), and that the effect is quite precisely estimated.

(1a) This finding appears to reflect the existence of a cluster of relatively small packages in which more than half of the code is remains uncredited. Rather strikingly, for the 10 packages in this group the proportion of unsigned code in the total number of bytes averages 70.7 percent; a similar uncredited proportion (averaging 72.5 percent) characterizes the five smallest packages within that group, whose average size was only 56 K-bytes. It seems plausible that in some of these cases there was only one developer who wrote such a large part of the code (averaging 40 K-bytes) that the entire package was identified with that contributor. In such cases, signing the contribution would have been redundant if gaining peer recognition for the work was a main motivation.

(1b) In support of the plausibility of the foregoing suggestion, it may be noted that inspection of the 8 packages in Linux kernel Version 2.5.25 where all the code was signed by one developer alone, the package sizes range between 9.8 and 40 K-bytes, the median lying at 18 K-bytes. Thus, it would not represent an extraordinarily large effort for a single individual to have contributed the unsigned portion of the small packages in which the preponderant part of the code was left uncredited ("unsigned").

(2) The estimates for **Equation 2** show that the number of developers contributing to a module is an increasing function of package size, and of the number of other modules that depend upon the package – reflecting its technical importance in the architecture of the Linux kernel. Both effects are statistically significant, although the size effect is more precisely estimated.

(2a) That larger packages should require the attention of a larger number of developers is certainly what would be expected from the software engineering perspective. But these estimates can be read to imply also that larger packages tend to attract more "productive" contributors, in the sense that the average amount of code contributed is significantly increasing with package size. The estimated elasticity of *numdev* w.r.t. *numbytes* is 0.615, implying that the estimated elasticity of (numbytes/numdev) is 0.385, significantly positive (about 3.5 times its standard error).

(2b) It should be stressed that (on the basis of the cross- section data studied here) it is not possible to identify which of the two likely processes generated the appearance of "increasing returns to scale." The wording of the foregoing is consistent with the possibility that what might account for the rise of average productivity with scale is that an "active" and growing module, in attracting the attention of developers also tends to attract developers with a capacity to contribute larger blocks of code. If it is known that more code is being added in a particular package, it is a fair inference that there will be a larger number of developers currently active at that sight. They represent a larger "potential peer audience," and simply on that account the site would be more attractive to developers who were able to make a larger contribution in terms of the amount of code they submitted. In other words, the developer population is heterogeneous in the code-writing capabilities of its members (at least within a limited time interval), and the dynamic "swarming" process at sites that become active is non-random in its selection among those capabilities. But, the hypothesized "selection effect" is not the only plausible interpretation for the statistical result. If the early contributions to a module established an architecture that made it feasible for subsequent individual contributors (with the same time input) to write more lines of good code (i.e., code that would be incorporated into and survive to be found in later releases), that would correspond to the engineering sense of increasing returns to scale. The architecture in this scenario constitutes the indivisibility, which accounts for the non-convexity of the production

function. Obviously there is nothing that would prevent both of the conjectured processes from operating concurrently in different package, or, indeed over the course of a given module's evolution.

(2c) There is a positive and statistically significant effect upon the number of developers participating in a package when that module's has a higher absolute dependency value – as measured by *depnum*, the number of packages upon which it depends ("calls"). This contrasts with the weakly negative, but statistically non-significant effect upon developer participation in cases where the package is of greater "technical criticality" in the Linux kernel – were that is gauged from its support value, *supnum* , the number of packages that "call" its code in order to function..

If these absolute measures of the directed connectivity were positively correlated, say because modules having more lines of code both called on, and were in turn called by a greater number of packages, we might expect to find that the product term (*supnum)x(depnum*) had picked up some of the positive effect of module size upon developer participation.. But there are some very large packages in the Linux kernel which occupy positions of lower "connectedness" – having comparatively small support – and dependency-values. This fact makes it more readily understandable that the product term's coefficient turns out to be statistically insignificant as well as quite small in magnitude.

What this tells us quite directly is that packages of a given size that could be viewed solely *on software engineering grounds* to be "less technically critical" are those that tend to attract contributions form a larger number of developers than would be accounted for by their size alone. Indirectly this result could be read as consistent with the view of some observers that while there is much esteem to be gained by making substantial contributions to a module that is critically connected with many others – that is has a high support value (*supnum*) – such work tends to go first to developers that have already acquired wide repute, either for early contributions made elsewhere in the project, or on different open source project altogether. In other words, for such technically important packages the entry standards (in terms of expertise and the magnitude of the effort required) in order to make substantial "commits" to such module can be set can be set higher, concentrating the critical code development tasks in fewer hands.

(2d) A first lesson that can be drawn from the preceding discussion echoes the economists' familiar chant about supply and demand: yes, there is a demand for opportunities to win fame, but the supply of attractive opportunities is not perfectly elasticity, so that strong demand tends to raise the "entry requirement" for those arena where the potential rewards for successful performance are especially high.

From the obverse of the same coin we may read a second equally simple message: for the mass of able developers who are acquiring new skills and experience, yet are not fabled talents, nor even versatile experts, comparatively strong attractions would be exerted by modules that were characterized by high dependency values, and generally low support values.[18]

[18] Of such packages Linux Version 2.5.25 has many to offer, primarily among its 30 network sub-projects and the 46 file system directories. Drivers also have comparatively high absolute dependency values vis-à-vis the modules belonging to other technical categoies, including those mentioned here. The *supnum* values typical of the 39 driver-packages in this version of the Linux kernel exceed their *depnum* values by factors between 2 and 3, but given the relative magnitude of the estimated coefficients for those variable reported for Equation 2 in Table 2.1 the net attraction exerted by the high "dependency-effects" are quite strong across the whole category of Drivers. It is not the case that large number of developers were engaged on all of these packages. Quite the contrary, there is very pronounced clustering on a few projects among those belonging to the ile system group,

A third point that is worth noting concerns the difference in the strength of the drive to gain recognition and peer approve, which we should expect would distinguish the member of the two groups of developers that have figured in the forgoing discussion. Among the already acclaimed programming "stars visible in the firmament of the open source software universe -- and even for some non-radiant bodies, who, planet-like, may be prominent enough to be seen from afar – the internal impulse to claim credit for each and every contribution is likely to have dissipated. Not so in the case of the striving young, and the comparative newcomers who have yet to possess the requisite skills to tackle big tasks in technically critical packages and projects. If it is they who are primarily draw with greater alacrity, and in larger numbers, by project-modules whose support value is low and whose dependency value is high, we might expect that they would join with a high propensity to sign whatever lines of code the project maintainer will allow them to contribute.

(3) This brings us to the estimates of **Equation 1.** The "log odds" – the natural logarithm of the ratio between uncredited and credited bytes – is essentially unaffected by the number of developers contributing to the package, but it is seen to vary positively with *supnum* and negatively with *depnum*. The latter pair of coefficients each is significantly different from zero at the 5 percent significance level on one-tail tests--being greater than zero and less than zero, respectively. Thus, the higher the support value of the module (supnum), the bigger is the ratio of unsigned to signed code; whereas that ratio shrinks as the dependency values of the modules rise. These results are consistent with the interpretation suggested for the respective signs of the coefficients of the same pair of variables in Equation 2 (see 2.b, above): the developers who make major contributions to packages that support many other modules are likely to include a substantial core of those very active, expert and highly estimated individuals, who have already gained the recognition of their peers (and the admiration of neophytes and journeymen programmers). On that count, they would be expected to exhibit a weaker propensity to seek to be credited for every line of code they contributed. By contrast, the ritual of code-signing would most likely be adhered to more assiduously among the greater numbers of the community. Virtually by definition, they, who are drawn to the projects where the dependency value is high, constitute the many that fame has thus far eluded. Some may still quest for peer esteem, while other take quiet satisfaction in having done a good piece of work that they are proud to claim, whereas still others find themselves move by practical circumstances to consider the value of establishing a recognized competence – with an eye to their future employment or self-employed in the software industry.

(3a) Neither of the technical variables'coefficients is estimated with very great precision, but their magnitudes are substantially different. For the same absolute increments in (change in the number of packages) in *supnum* and *depnum*, the positive effect on the proportion of code that is credited (*signed)* would be much bigger for the case of a rise of the

---

resulting in a very high degree of concentration of the 1200 or so developer contributions made in this technical category: 64% occurred in a more 6 % of the packages. The corresponding concentration of code was still more pronounced, which is what our findings about the increase in average developer productivity with project code size: the same 6 % among the file system modules absorbed 77.4% of the bytes written in the entire category. The phenomenon of concentration is quite ubiquitous across success releases of the Linux kernel, and appears in a wide swatch of open source software products. It is one among several meso-level features that we would wish to have replicated by the results of our stochastic simulation model.

dependency measure than would be the case for a fall in the support value. This follows obviously from a comparison of the two coefficients in Equation 3, the latter of which is a fraction (0.435) of the size of the estimated coefficient on *depnum.* An obvious implication is that these two "technical characteristics" of the modules in Linux 2.5.25 modules would be in approximate balance – and therefore without net effects on code-signing propensities – where the ratio *depnum: supnum* was in the neighborhood of 2.3. But, interestingly enough, that situation obtains only for a very few packages of the Linux kernel. For the rest, a substantial minority have *depnum/supnum* ratios well below unity, indeed in the neighborhood of 0.1; whereas the others' ratios are clustered around 3. One way to read these results is this: The impacts of the technical attributes of the modules upon their relative attractiveness to developers with differing degrees of motivation to seek peer recognition of their contributions is a factor that creates substantial inter-module variance in the proportion of code that is signed, but its overall effect is that of lowering the proportion of code that remains unsigned.

(3c) The number of developers contributing to the package exerts no appreciable independent effect on the probability that code is signed (credited). This result is not entirely unexpected, in light of the results already discussed in connections with Equation 2. If it was thought that an enlarged audience of "spectators" would induce a larger proportion of code to be signed in the expectation of gaining greater "peer regard," that would pre-suppose that there had been an exogenous increase in the sized of the relevant audience -- represented by the total number of developers engaged in contributing to the module in question. But the previous discussion suggested the possibility that increasing returns to code size might work via a selection effect to raise average productivity, there by checking the growth in the number of developers. If considerations of peer regard underlay the hypothesize bias in the selection effect, not only do we have to recognize the endogeneity of the total number of developers, but it would appear that peer regard would not work also exert an additional indirect effect via that channel on the developers' code signing propensities.

3.2 Summary of the Interpretative Discussion

Two main substantive points have emerged from the foregoing discussion and are immediately pertinent to the simulation exercises undertaken in the following section of the paper. The first lends a measure of independent and objective support for the credence that we place in the view of developers' behavior as being shaped at the *margin* by a mixture of psychological and material incentives formed in "the economy of regard." The indirect indications of the influence of considerations of "peer regard" upon code-signing behavior remain very general in character at this stage of our research. We still are a considerable distance from devising and econometrically implementing tests of the concrete and simplified propositions that the simulation model described in Section 3 has drawn from Raymond's (1999) stimulating observations about the specific structure of "prestige" and "personal achievement scores" that shape developers choices about how to allocate their efforts. For the purposes of advancing the simulation modeling project, we are willing to treat these "decision rules as maintained hypotheses, justifying that approach on the strength of the analogy between their putative role in the world of "open source" production, and the collegiate reputational reward system's role in guiding resource allocation within the sphere of "open science."

The second point is more straightforward, or if you like, "less tortured." It follows immediately as an implication finding that there is (significantly) increasing average

productivity with increases in code scale. An "active module" in a major projects, like an active project among the population of projects on sites such as SourceForge, would attract new developers at a higher rate per K-byte than is likely to do when the size of its code has grown much bigger (in K-bytes, or in thousand (physical) single lines of code, K-SLOCs). Hence, in early phases of a new module's development vis-à-vis that taking place in other modules, it would appear that the clustering of larger numbers at that site was an attractor of new contributors. Although our discussion indicates that there are other underlying explanations available for the observable phenomenon, an we prefer one or more of them to the view that there is a "causal" feedback mechanism at work, it may nevertheless be useful at this stage in our stimulation work to try to capture the apparent effect of numbers of recently active developers in raising the attractiveness that the module-project holds for new contributors. At some subsequent stage, however, it may be appropriate to introduce the complications needed to capture the effects of changes in the strength of the selection effects as the project matures.

3.3 Broader Implications

Insert: significance for modelling within project allocation.

The results emerging from this line of research also carry a general methodological message. They demonstrate the potentialities of using quantitative information extracted from the open source code itself, in order to illuminate the issue of developer's motives without having recourse to first hand knowledge gained by participant observers. Surmises on the part of the latter, like the 'insights' supplied to ethnographers by local informants, are not always accurate in representing the *mores*, motives and perceptions of the mass of "non-informative" participants. Behavioral evidence of the sort examined here, however limited in scope, and however qualified are the inferences to which it leads, should therefore supplement the analysis of survey responses.

REFERENCES

Arora, Seema, Paul.A. David and A. Waterman. 2003. "Interim Results from *FLOSS-US*, the 2003 Web-based Survey of Free and Open Source Developers," SIEPR-Project NOSTRA Working Paper (25th February).

Bezroukov, N. 1999. 'Open Source Software Development as a Special Type of Academic Research (Critique of Vulgar Raymondism),' *First Monday.* (http://www.firstmonday.dk/issues/issue4_10/bezroukov/index.html): Accessed 1 Feb. 03.

Bonaccorsi, Andrea and Cristina Rossi. 2003a. "Why Open Source software can succeed." *Research Policy* 32 (7):1243-1258 (July) [Special Issue on "Open Source Software Development", Edited by Eric von Hippel and Georg von Krogh.

———2003b. "Licensing schemes in the production and distribution of open source software: An empirical investigation." http://opensource.mit.edu/papers/bnaccorsirossilicense.pdf

———2003c. "Altruistic Individuals, selfish firms? The structure of motivation in open source software." http://opensource.mit.edu/papers/bnaccorsirossimotivationshort.pdf

———2003d**.** “Contributing to the common pool resources in open source software: A comparison between individuals and firms.” http://opensource.mit.edu/papers/bnaccorsirossidevelopers.pdf

———2003e. “Comparing motivations of individual programmers and firms to take part in the open source movement: From community to business.” http://opensource.mit.edu/papers/bnaccorsirossimotivationlong.pdf

Boston Consulting Group. 2002. Survey of free software/open source developers conducted by the Boston Consulting Group. See www.osdn.com/bcg

Brooks, Frederick P. 1995. *The Mythical Man-Month: Essays on Software Engineering,* Anniversary Edition of the 1982 volume. Addison Wesley Publishing Company.

Carayol, Nicolas and Jean-Michel Dalle. 2000. “Science wells: Modelling the ‘problem of problem choice’ within scientific communities.” Presented at the 5th WEHIA Conference, GREQAM, Marseille, June.

Comino, Stefano and Fabio M. Manenti. 2003. “Open Source vs Closed Source Software: Public Policies in the Software Market.” http://opensource.mit.edu/papers/cominomanenti.pdf

Cowan, Robin and Elad Harison. 2004. “On Substitution of Intellectual Property and Free Disclosure: An Analysis of R&D Strategies In Software Technologies.” Forthcoming, *Economics of Innovation and New Technology*.

Dalle, Jean-Michel and Paul A. David. 2001. “On open source software and the organization of cathedral-building: metaphors and realities.” Working Paper, SIEPR-NOSTRA Project on the Economics of Open Source Software, December.

_____. 2003. “The Allocation of Software Development Resources in ‘Open Source’ Production Mode,” SIEPR-Project NOSTRA Working Paper, (15th February).

Dalle, Jean-Michel Dalle and Paul A. David (2004). “The Allocation of Software Development Resources in ‘Open Source’ Production Mode,” Forthcoming in Joe Feller, Brian Fitzgerald, Scott Hissam and Karim Lakhani, *Making Sense of the Bazaar,* Cambridge, MA: MIT Press, 2004.

Dalle, Jean-Michel, Paul A. David and W.E. Steinmueller. 2002. “An Agenda for Integrated Research on the Economic Organization & Efficiency of F/LOSS Software Production.” Available at: http://siepr.stanford.edu/programs/OpenSoftware_David/FLOSS%20Conf%20Stmt_JMD+PD+ES_v6.htm

Dalle, Jean-Michel and Nicolas Jullien. 2000. “NT vs. Linux, or some explorations into the economics of free software,” In: *Application of simulation to social sciences*, G. Ballot and G. Weisbuch, eds. Paris, France: Hermès, pp. 399-416.

Dalle, Jean-Michel and Nicolas Jullien. 2003. “ ‘Libre’ software : turning fads into institutions?”, *Research Policy*, 32(1):1-11.

Dasgupta, Partha and Paul A. David. 1987. Information Disclosure and the Economics of Science and Technology. ch. 16 in *Arrow and the Ascent of Modern Economic Theory*, (G. Feiwel, ed.), New York: New York University Press, 1987, pp. 519-542.

———1994. “Toward a new economics of science”, *Research Policy*, vol. 23, no. 5, pp. 487-521.

——— 1998a. Communication Norms and the Collective Cognitive Performance of ‘Invisible Colleges in *Creation and Transfer of Knowledge: Institutions and Incentives,* Physica-Verlag Series *Contributions to Economics*, G.Barba. Navaretii et al., eds., Berlin, Heidelberg, New York: Springer-Verlag.

——— 1998b. "Reputation and Agency in the Historical Emergence of the Institutions of ‘Open Science’," *Center for Economic Policy Research, Publication No. 261*, Stanford University, (revised March 1994), further revised :December.

——— 1998c. Common Agency Contracting and the Emergence of ‘Open Science’ Institutions, *American Economic Review*, 88(2): 15-21 (May).

——— 2000. “Patronage, Reputation, and Common Agency Contracting in the Scientific Revolution: From Keeping ‘Nature’s Secrets’ to the Institutionalization of ‘Open Science.” (Unpublished; under review at *Journal of Economic History*).

———2001. “Path dependence, its critics and the quest for ‘historical economics’,” in *Evolution and Path Dependence in Economic Ideas: Past and Present*, eds. P. Garrouste and S. Ioannidies. Cheltenham, Glos.: Edward Elgar, 2001.

David**,** Paul A., Seema Arora and W. Edward Steinmueller. 2001. “Economic Organization and Viability of Open Source Software: A Proposal to The National Science Foundation,” SIEPR, Stanford University, 22 January.

DiBona, Chris, Sam Ockman, and Mark Stone. 1999. “Introduction,”In: *Open Sources: Voices from the Open Source Revolution*, C. DiBona, S. Ockman, and M. Stone, eds. Sebastopol, Calif.: O'Reilly & Associates, pp. 1-17.

Franke, Nikolaus and Eric von Hippel. 2002. “Satisfying heterogeneous user needs via innovation toolkits: the case of Apache security software,” MIT Sloan School of Management Working Paper No. 4341-02, January.

Gambardella, Alfonso and Bronwyn H. Hall. 2004.”Proprietary vs. Public Domain Licensing of Software and Research Products.” Working Paper. Scoula Superiore Sant’ Anna, Pisa. February. (Revised version forthcoming in *Research Policy*.

German, D. M. 2002. “The Evolution of the GNOME Project,” Workshop Proceeding presented at ‘Meeting Challenges and Surviving Success: The 2nd Workshop on Open Source Software Engineering’ (May 19-25), Available at http://opensource.ucc.ie/icse2002/German.pdf.

Ghosh, Rishab Aiyer. 1998. “Cooking pot markets: an economic model for the trade in free goods and services on the Internet,” *First Monday,* 3(3) http://www.firstmonday.org/issues/issue3_3/ghosh/index.html

Ghosh, Rishab Aiyer. 2003. “Clustering and Dependencies in Free/Open Software Development: Methodology and Preliminary Analysis,” MERIT-Infonomics Institute and SIEPR-Project NOSTRA Working Paper (First version: June 2002; revised: 15th February).

Ghosh, Rishab Aiyer and Paul A. David. 2003. “The nature and composition of the Linux kernel developer community: a Dynamic Analysis,” SIEPR-Project NOSTRA Working Paper (21st February).

Ghosh, Rishab Aiyer, Rudiger Glott, Bernhard Kreiger and Gregario Robles. 2002. *The Free/Libre and Open Source Software Developers Survey and Study—FLOSS Final Report.* June. http://www.infonomics.nl/FLOSS/report/

González-Barahona, Jesús M. et al. 2002. “Counting potatoes: The size of Debian 2.2,” (Version 3a: 3 January). http://people.debian.org/~jgb/debian-counting/counting-potatoes/.

González-Baharona, Jesus M., Luiz Lopez and Gregorio Robles. 2004. "The community structure of the modules in the Apache project." GSyC Working Paper, Universidad Rey Juan Carlos (Mostoles). February.

Harhoff, Dietmar, J. Henkel and Eric von Hippel. 2000. “Profiting from Voluntary Information Spillovers: How Users Benefit by Freely Revealing their Innovations.” (July). opensource.mit.edu/papers/evhippel-voluntaryinfospillover.pdf.

Kelty, Christopher M. 2001. “Free Software/Free Science.” *First Monday,* December. www.firstmonday.org/issues/issue6_12/kelty/index.html.

Koch, S. and G. Schneider. 2000. “Results From Software Engineering Research Into Open Source Development Projects Using Public Data,” Vienna University of Economics and Business Administration http://opensource.mit.edu/papers/koch-ossoftwareengineering.pdf

Kogut, B. and A. Metiu. 2001. “Open-Source Software Development and Distributed Innovation,” *Oxford Review of Economic Policy* 17 (2): 248-64.

Krishnamurthy, S. 2002. “Cave or Community? An Empirical Examination of 100 Mature Open Source Projects,” University of Washington, Bothell. May. http://opensource.mit.edu/papers/krishnamurthy.pdf.

Kuan, Jennifer. 2001. “Open Source Software as Consumer Integration into Production,” http://opensource.mit.edu/papers/Jenny%20Kuan%20-%20Open%20Source%20Software%20As%20Integration%20into%20Production.pdf.

Kuwabara, Ko. 2000. “Linux: A Bazaar at the Edge of Chaos,” *First Monday*, 5:3 (March). firstmonday.org/issues/issue5_3/kuwabara/index.html

Lerner, Josh and Jean Tirole. 2002. “The Simple Economics of Open Source.” National Bureau of Economic Research (NBER) Working Paper 7600 (March). www.nber.org/papers/w7600.

Lerner, Joshua and Jean Tirole. 2003. ”The Scope of Open Source Licensing.” http://opensource.mit.edu/papers/lernertirole2.pdf

Madey, G., V. Freeh and R. Tynan (2002). “*The Open Source Software Development Phenomenon: An Analysis Based on Social Network Theory”. Proceedings Americas Conference on Information Systems* (AMCIS2002), Dallas, TX.

Mateos-Garcia, J. and W. E. Steinmueller. 2003a. "The Open Source Way of Working: A New Paradigm for the Division of Labour in Software Development?" Falmer, UK, SPRU -- Science and Technology Policy Research, INK Open Source Working Paper No. 1. January.

———2003b. “Dynamic Features of Open Source Development Communities and Community Processes,” Brighton: SPRU -- Science and Technology Policy Studies, Open Source Movement Research INK Working Paper No. 3. February.

Mauss, M., *The gift: the form and reason for exchange in archaic societies. (Transl. W. D. Halls), 1925, republished 1990.*

McGowan, David (2001) “Legal Aspects of Free and Open Source Software,” University of Illinois Law Review, 241. [See the author’s forthcoming adaption of this article, under the same title, forthcomig as Ch. 24 in Joe Feller, Brian Fitzgerald, Scott Hissam and Karim Lakhani, *Making Sense of the Bazaar,* Cambridge, MA: MIT Press, 2004].

Merton, Robert K. 1973. *The Sociology of Science: Theoretical and Empirical Investigations.* Edited and with an introduction by Norman W. Storer. Chicago: University of Chicago Press.

Nichols, D. and M. Twidale. 2003. "The Usability of Open Source Software,"*First Monday.* (http://firstmonday.org/issues/issue8_1/nichols/index.html): Last Accessed 5 February 03.

Offer, Avner. 1997. “Between the Gift and the Market: The Economy of Regard”, *Economic History Review*, vol. 50, 3 (Aug. 1997), pp. 450-476.

Raymond, Eric S. 1998a. "The Cathedral and the Bazaar," *First Monday*, volume 3, number 3 (March), firstmonday.org/issues/issue3_3/raymond/index.html and www.tuxedo.org/~esr/writings/cathedral-bazaar.

———1998b. "Homesteading the Noosphere," *First Monday*, 3: 10 (October), firstmonday.org/issues/issue3_10/raymond/index.html and www.tuxedo.org/~esr/writings/homesteading.

———1999a. “A Response to Nikolai Bezroukov, ” *First Monday*, 4:11 (November 1999). firstmonday.org/issues/issue4_11/raymond/index.html.

_____2001.*The Cathedral and the Bazaar: Musings on Linux and Open Source by an Accidental Revolutionary*. Sebastapol, CA: O’Reilly.

Reagle, Jr. Joseph M. 2003. “Socialization in Open Technical Communities.” http://reagle.org/joseph/2003/socialization/voluntary.html

Rheingold, Howard. 1993. *The Virtual Community: Homesteading on the Electronic Frontier*. Reading MA: Addison-Wesley Publishing Company.

Robles, Gregorio and Rishab A. Ghosh. 2004. “CODD (source code) Dependency Analyzer (codd-dependency." MERIT Working Paper [Presented to the *OWLS* Workshop, held at Oxford Internet Institute, Oxford 25-26 June 2004].

Shah, Sonali. 2003. “Understanding the Nature of Participation and Coordination in Open and Gated Source Software Development Communities.” Chapter 4 of dissertation. http://opensource.mit.edu/papers/shah3.pdf

Tapscott, D, D Ticoll, and A Lowy. 2000. *Digital Capital: Harnessing the Power of Business Webs*. Cambridge MA: Harvard Business School Press.

von Hippel, Eric. 2002. "Horizontal innovation networks - by and for users" Cambridge, MA, Massachussetts Institute of Technology, Sloan School of Management, Working Paper No. 4366-02. June.

Waterman, Andrew H. 2003. “Why They Do It and What that Implies for the Boundaries of Open Source Development Projects,” SIEPR-Project NOSTRA Working Paper. (First draft: November 2002. Revision: February)

SIEPR Open Source Software Project Working Papers

Note: Abstracts of all the papers listed here, and full texts of selected items Papers are available at: http://siepr.stanford.edu/programs/OpenSoftware_David/NSFOSF_Publications.html.

Arora, Seema, Paul A. David and Andrew H. Waterman (2003). "Interim Results from *FLOSS-US*, the 2003 Web-based Survey of Free and Open Source Developers," SIEPR-Project NOSTRA Working Paper (February).

Arora, Seema and Paul A. David (2003). "Commercialization of Open Source Software: Symbiotic or Parasitic?," SIEPR-Project NOSTRA Working Paper. (8 February; revised September). [Presently under revision for submission to the *Harvard Business Review*].

Dalle, Jean-Michel Dalle and Paul A. David (2003). "The Allocation of Software Development Resources in 'Open Source' Production Mode," SIEPR-Project NOSTRA Working Paper, (February 15). [Accepted for publication in Joe Feller, Brian Fitzgerald, Scott Hissam and Karim Lakhani, *Making Sense of the Bazaar,* forthcoming from MIT Press, 2004.]

David, Paul A. and Rishab Aiyer Ghosh (2003), "Free and Open Source Software Developers and "the Economy of Regard: A Quantitative Analysis of Code-Signing Patterns within the Linux Kernel," SIEPR-Project NOSTRA Working Paper, (February 22).

David, Paul A., Andrew H. Waterman and Seema Arora (2003). "FLOSS-US: The Free/Libre Open Source Software Developer Survey for 2003: A First Report." (September) [Available at: http://www.stanford.edu/group/floss-us/report/FLOSS-US-Report.pdf.]

Juan Mateos Garcia and W. Edward Steinmueller (2003a), "The Open Source Way of Working: A New Paradigm for the Division of Labour in Software Development?," INK Open Source Research Working Paper No. 1, University of Sussex-SPRU (January).

Juan Mateos Garcia and W. Edward Steinmueller (2003b), "Aplying the Open Source Development Model to Knowledge Work," INK Open Source Research Working Paper No. 2, University of Sussex-SPRU (January).

Rishab Aiyer Ghosh (2003), "Clustering and Dependencies in Free/Open Software Development: Methodology and Preliminary Analysis," MERIT-Infonomics Institute and SIEPR-Project NOSTRA Working Paper (First version: June 2002; revised: February 15). [Accepted for publication in Joe Feller, Brian Fitzgerald, Scott Hissam, Karim Lakhani, eds., *Making Sense of the Bazaar,* forthcoming from MIT Press in 2004..]

Rishab Aiyer Ghosh and Paul A. David (2003), "The nature and composition of the Linux kernel developer community: a Dynamic Analysis," SIEPR-Project NOSTRA Working Paper (February 21).

Steinmueller, W. Edward (2003), "Exploring the limits of the 'open source' production mode: differences in the value of collective effort in knowledge-creation activities when coordination requirements vary," INK Open Source Research Working Paper No. 3, University of Sussex-SPRU, August.

Waterman, Andrew H. (2003), "Why They Do It and What that Implies for the Boundaries of Open Source Development Projects," SIEPR-Project NOSTRA Working Paper. (First draft: November 2002. Revision: February).