Simulating Code Growth in Libre (Open-Source) Mode

By

Jean-Michel Dalle
University Pierre-et-Marie-Curie & IMRI-Dauphine
Jean-Michel.Dalle@upmc.fr

&

Paul A. David
Stanford University & Oxford Internet Institute
pad@stanford.edu

First draft: 27 December 2004
This version: 31 January 2005

ACKNOWLEDGEMENTS

The research reported in this paper was made possible by grant awards to the Stanford University (SIEPR) Project on the Economics of Free & Open Source Software and its European academic partners by the National Science Foundation program on Digital Technology and Society: IIS-0112962 (2001-04) and IIS-0329259 (2003-05). [http://siepr.stanford.edu/programs/OpenSoftware_David/OS_Project_Funded_Announcmt.htm.] Our research has also been supported by *CALIBRE*, a EU FP6 Coordination Action. We have benefited from discussions with participants at the EPIP3 and OWLS seminars, which convened at Scuola Superiore Sant'Anna in Pisa, Italy, on 2-3 April 2004, and at the Oxford Internet Institute in Oxford, UK, on 25-26 June 2004, respectively. Fabio Arcangeli, Robin Cowan, Brian Fitzgerald, Alfonso Gambardella, Rishab A. Ghosh, Jesus Gonzales-Baharona, Bronwyn H. Hall, Jim Herbsleb, Eric von Hippel, Brian Kahin, Mathieu Lacage, Karim Lakhani, Juan Mateos-Garcia, Stephen M. Maurer, Jean-Charles Pomerol, Gregorio Robles, Walt Scacchi, and Ed Steinmuller all had a helpful hand in shaping this work. But, the views expressed and the defects that remain are ours.

Simulating Code Growth in Libre (Open-Source) Mode

Jean-Michel Dalle and Paul A. David

SUMMARY

We present an original modeling tool that can be used to study the social mechanisms by which individual software developers' efforts are allocated within large and complex open source projects. The dynamical agent-based model is first described analytically in a deterministic discrete choice framework. Next, the results of simulations experiments using a stochastic specification are presented, to study the effects of various structural parameters that reflect "community norms" and governance rules affecting the behaviors of individuals associated with the particular project. In addition to the relative peer evaluation of different kinds of programming work associated with its constituent modules, individual developer's behaviors choices among the latter appear to be affected by the clustering of others developers at certain "hot spots" of development activity. Allowance for that effect enables the simulations to generate the very high Gini coefficients describing the empirical distributions of modules sizes that are reported in the literature.

The model is dynamic, with contributions of code being added sequentially either to existing modules, or to create new modules that are technically related to existing ones: consequently, the emerging global architecture of the project's code can be conveniently represented as an evolving hierarchical tree. For a particular "tree'' its morphological features at any given moment in time affects both the interest it holds for developers at that moment, and its utility in application by end-users. Introducing a simple representation of the latter agents' social utility function, we find preliminary but striking indications that the social "reward structure" (that sociological observers suggest is influential in developers' decisions about what and where to contribute) is not particularly not well aligned to produce an eventual simulated code architecture that approaches optimally when evaluated from the viewpoint of end-users. This may have some significance for those who stress the purely "self-organized" and undirected features of open source software projects. When allowance is made for the existence of project governance rules, however, it is found that the social utility index of the eventual code-tree can be enhanced substantially if maintainers' policies set minimum standards for "commits" that nonetheless facilitate the "early release" of successive versions of the code.

1. Introduction

The initial contributions to the social science literature addressing the phenomenon of Libre (open-source, free) software have been directed primarily to identifying the motivations underlying the sustained and often intensive engagement of many highly skilled individuals in this non-contractual and unremunerated mode of production.[1] That focus reflects a view that widespread voluntary participation in the creation and free distribution of economically valuable goods is something of an anomaly, at least from the viewpoint of mainstream microeconomic analysis.

[1] See, among the salient early contributions to the "economics of open source software," Ghosh (1998), Harhoff, Henkel and von Hippel (2000), Lakhani and von Hippel (2000), Lerner and Tirole (2000), Weber (2000), Kogut and Metiu (2001).

A second problem that has occupied observers, and especially economists, is to uncover the explanation for the evident success of products of the Libre software mode in market competition against proprietary software – significantly on the basis not only of their lower cost, but their reputedly superior quality.[2] This quest resembles the first, in reflecting a state of surprise and puzzlement about the apparently greater efficiency that these voluntary, distributed production organizations have been able to attain *vis-à-vis* centrally-managed, profit-driven firms that are experienced in creating "closed," software products.

Anomalies are intrinsically captivating for intellectuals of a scientific, or just a puzzle-solving bent. Yet, the research attention that has been stimulated by the rapid rise of a Libre software segment of the world's software-producing activities during the 1990's owes something also to the belief that this phenomenon and its relationship to the free and open software movements, could turn out to be of considerably broader social and economic significance. There is, indeed, much about these developments that remains far from transparent, and we are sympathetic to the view that a deeper understanding of them may carry implications of a more general nature concerning the organization of economic activities in networked digital technology environments. Of course, the same might well be said about other aspects of the workings of modern economies that are no less likely to turn out to be important for human well-being.

Were the intense research interest that Libre software production currently attracts to be justified on other grounds, especially as a response to the novelty and mysteriousness of the phenomena, one would need to point out that this too is a less than compelling rationale; the emergence of Libre software activities at their present scale is hardly so puzzling or aberrant a development as to warrant such attention. Cooperative production of information and knowledge, among members of distributed epistemic communities who do not expect direct remuneration for their efforts simply cannot qualify as a new departure. There are numerous historical precursors and precedents for Libre software, perhaps most notably in the "invisible colleges" that appeared among the practitioners of the new experimental and mathematically approaches to scientific inquiry in western Europe in the course of the 17th century.[3] The professionalization of scientific research, as is well known, was a comparatively late development, and, as rapidly as it has proceeded, it has not entirely eliminated the contributions of non-professionals in some fields, optical astronomy being especially notable in this regard; communities of "amateur" comet-watchers persist, and their members continue to score – and to verify – the occasional observational coup.

"Open science," the mode of inquiry that emerged and became formally institutionalized during the era of the Scientific Revolution under systems of public and private patronage thus offers an obvious cultural and organizational point of reference for observers of contemporary communities of programmers engaged in developing free software and open source software.[4] The "communal" ethos and norms of "the Republic of Science"

---

[2] In this particular vein, see, for example Dalle and Jullien (2000, 2003), Bessen (2001), Kuan (2001), Benkler (2002).

[3] See, e.g., David (1998a, 1998b, 2001, 2004) and references to the history of science literature supplied therein.

[4] This has not gone unrecognized by observers of the free and open software movements. In "The Magic Cauldron," Raymond (1999c) explicitly notices the connection between the information-sharing behavior of academic researchers and the practices of participants in Libre projects. Further, Raymond's (1998b) illuminating discussion of the norms and reward systems (which motivate and guide developers selections of projects on which to work) quite clearly parallels the classic approach of Robert K. Merton (1973) and his followers in the sociology of science. This is underscored by Raymond's (1999b) rejoinder to N. Berzoukov's (1999) allegations on the point. See also DiBona et al. (1999) for another early discussion; Kelty (2001), and

emphasize the cooperative character of the larger purpose in which individual researchers are engaged, stressing that the accumulation of reliable knowledge is an essentially social process. The force of its universalistic norm is to render entry into scientific work and discourse open to all persons of "competence," while a second key aspect of "openness" is promoted by norms concerning the sharing of knowledge in regard to new findings and the methods whereby they were obtained.

Moreover, a substantial body of analysis by philosophers of science and epistemologists, as well as theoretical and empirical studies in the economics of knowledge, points to the superior efficiency of cooperative knowledge-sharing among peers as a mode of generating additions to the stock of scientifically reliable propositions.[5] In brief, the norm of openness is incentive compatible with a collegiate reputational reward system based upon accepted claims to priority; it also is conducive to individual strategy choices whose collective outcome reduces excess duplication of research efforts, and enlarges the domain of informational complementarities. This brings socially beneficial spill-overs among research programs and abets rapid replication and swift validation of novel discoveries. The advantages of treating new findings as *public goods* in order to promote the faster growth of the stock of knowledge are thus contrasted with the requirement of restricting informational access in order to enlarge the flow of privately appropriable rents from knowledge stocks.

The foregoing functional juxtaposition suggests a logical basis for the existence and perpetuation of institutional and cultural separations between two normatively differentiated communities of research practice. The open "Republic of Science" and the proprietary "Realm of Technology" on this view, constitute distinctive organizational regimes each of which serves a different (and potentially complementary) societal purpose. One might venture farther to point out that the effective fulfilling of their distinctive and mutually supporting purposes was for some time abetted by the ideological reinforcement of a normative separation between the two communities; by the emergence of a distinctive ethos of "independence" and personal disinterested-ness ("purity") that sought to keep scientific inquiry free to the fullest extent possible from the constraints and distorting influences to which commercially-oriented research was held to be subject.

It follows that if we are seeing something really new and different in the Libre software phenomenon, that hardly can inhere in attributes shared with long-existing open science communities.[6] Rather, it must be found elsewhere, perhaps in among the more

---

David, Arora and Steinmueller (2001), expand the comparison with the norms and institutions of open/academic science. Nevertheless, one should observe that that the parallel is by no means exact: formal professional accreditation and institutional affiliation is a salient *de facto* requirement for active participation in modern academic and public sector research communities, yet the computer programming and other software development tasks – whether in the commercial or the free and open source spheres -- remains a an activity that has resisted becoming "professionalized."

[5] See Dasgupta and David (1994), David (1998b, 2002a, b) on the cognitive performance of open science networks in comparison with that of proprietary research organizations; David (2003) on the interaction between modern 'open science' and proprietary R&D.

[6] The phenomenon of free and open source software is perceived by Benkler (2002: pp. 1-2) as an exemplifying "a much broader social-economic phenomenon.…the broad and deep emergence of a new, third mode of production in the digitally networked environment." This mode he labels "'commons-based peer production', to distinguish it from the property- and contract-based modes of firms and markets. Its central characteristic is that groups of individuals successfully collaborate on large scale projects following a diverse cluster of motivational drives and social signals rather than either market prices or managerial commands." Anyone at all familiar with the history of open science since the 17th century will be disconcerted – to say the least --by this particular imputation of novelty and significance to Libre projects.

distinctive items in the following list: (a) the sheer scale on which these activities are being conducted, (b) the global dispersion and heterogeneous backgrounds of the participants (and the related absence of mandatory professional certification requirements), (c) the rapidity of their transactions, and (d) the pace at which their collective efforts reach fruition. This shift in conceptualization has the effect of turning attention to a constellation of technical conditions whose coalescence has especially affected this field of endeavor. Consider just these three: the distinctive immateriality of "code," the great scope for modularity in the construction of software systems, and the enabling effects of advances in digital (computer-mediated) telecommunications during the past several decades. Although it might be thought that the intention here is merely to portray the historically unprecedented features of the Libre software movements as primarily an "Internet phenomenon," we have something less glib than that in mind.

It is true that resulting technical characteristics of both the work-product and work-process alone cannot be held to radically distinguish the creation of software from other fields of intellectual and cultural production in the modern world. Nevertheless, they do suggest several respects in which it is misleading to interpret the Libre software phenomenon simply as "another sub-species of 'open science'." The knowledge incorporated in software differs in at least two significant respects from the codified knowledge typically produced by scientific and technical work groups. Computer software is a form of information that is at the same time a "technological artifact," which is to say that it has immediate functional effectiveness without requiring further expenditures of effort upon development.[7] This immediacy has significant implications not only at the micro-level of individual motivation, but for the dynamics of collective knowledge-production. Indeed, because software code is "a machine implemented as text," its functionality is peculiarly self-exemplifying. Thus, "running code" serves to short-circuit many issues of "authority" and "legitimation" that traditionally have absorbed much of the time and attention of scientific communities; and to radically compress the processes of validating and interpreting new contributions to the stock knowledge.[8]

In our view Libre software production activities warrants systematic investigation not as a sub-species of the unusual class of technological objects called "computer software," but because its relationship with a conjunction of a particular set of trends in the modern economy may give this development significant implications for the future of the advance of knowledge, and consequently for knowledge-driven economic growth. The first of those trends is that information-goods that share many of the special properties of software have been moving more and more to center-stage among the drivers of sustainable economic development. Secondly, the enabling of peer-to-peer organizations for information distribution and utilization is an increasingly obtrusive consequence of the direction in which digital technologies are advancing. Thirdly, the "open" (and cooperative) mode of organizing the generation of new knowledge has long been recognized to have efficiency properties that are much superior to institutional solutions to the public goods problem that entail the restriction of access to information through secrecy or property rights enforcement, but to pose a problem inasmuch as it seemingly requires a rising volume of public funding for "basic research. Fourthly, and of practical significance for those who seek to study it systematically, the Libre software mode of production itself is generating a wealth of

---

[7] This property of software, incidentally accounts for its anomalous treatment under intellectual property law. Software, being "a machine" implemented as "text," is unique in being both patentable and copyrightable.

[8] Therefore, at the risk of re-circulating a tired bromide, it might well be said that in regard to the sociology and politics of the open source software communities, "the medium is the message."

quantitative information about this instantiation of "open epistemic communities." This latter development makes Libre software activities a valuable window through which to study the more generic and fundamental processes that are responsible for its power, as well as the factors that are likely to limit its domain of viability in competition with other modes of organizing economic activities.

Proceeding from this re-framing of the phenomenon, one is led toward a conceptual approach that highlights a broader, ultimately more policy-oriented set of issues than those which hitherto have dominated the economics literature concerning Libre software. A correspondingly re-orientation of research agendas would appear to be called for.[9] Its analytical elements are in no way novel, however, but merely newly adapted to suit the subject at hand. It is directed to answering a fundamental and interrelated pair of questions: First, by what mechanisms do Libre software projects mobilize the human resources, allocate the participants diverse expertise, coordinate the contributions and retain the commitment of their members? Second, how fully do the products of these essentially self-directed efforts meet the long-term needs of software users in the larger society, and not simply provide satisfactions of various kinds for the developers? These will be recognized immediately by economists to be utterly familiar and straightforward – save for not yet having been explicitly posed or systematically pursued in this context.

Pursuing these concrete, classic economic questions compels a detailed examination of the actual workings of the system of social organization that actually allocates software development resources among the various software systems and applications projects that are being undertaken by "communities" of distributed and sometimes anonymous volunteers -- as it is the situation of the large projects are found in the world of Libre software. How does the ensemble of developers collectively "select" among the observed array of projects that are launched? What processes govern the mobilization of sufficient resource inputs to enable some among those to attain the stage of functionality and reliability that permits their being diffused into wider use – that is to say, use beyond the circle of programmers immediately engaged in the continuing development and 'debugging' of the code itself?

Indeed, it seems only natural to expect that economists would provide an answer to the question of how, in the absence of directly discernible market links between the producing entities and "customers," the output mix of the open source sector of the software industry is determined. Yet, surprisingly, this question does not appear to have attracted any significant amount of attention. This curious lacuna, moreover, is not a deficiency peculiar to the economics literature, for, it is notable also in the writings of some of the Libre software movement's pioneering participants and popular exponents.[10] Although enthusiasts have made numerous claims regarding the qualitative superiority of products of the open source mode when these are compared with software systems tools and applications packages developed by managed commercial projects, scarcely any attention is directed to the issue of

---

[9] This is the approach being pursued by the members of the project on The Economic Organization and Viability of Open Source Software at Stanford University and its research partners at academic institutions in France, the Netherlands and Britain. Most of the researchers associated with this project come to this particular subject matter from the perspective formed by their previous and on-going work in "the new economics of science," which has focused attention upon the organization of collaborative inquiry in the "open science" mode, the behavioral norms and reinforcing reward systems that structured the allocation of resources, the relationships of these self-organizing and relatively autonomous epistemic communities with their patrons and sponsors in the public and private sectors. See Dalle, David and Steinmueller (2002) for the scope of this integrated research agenda.

[10] See, e.g., Raymond (1998b, 1999); Stallman (1999), Dibona, Ockman and Stone (1999) and the statements of contributors collected therein.

whether the array of completed Libre software projects also is "better" or "just as good" in responding to the varied demands of software users.

It is emblematic of this gap that the metaphor of "the bazaar" was chosen by Eric S. Raymond (1998a) to convey the distinctively un-managed, decentralized mode of organization that characterizes open source software development projects. Here is a representative reading of this aspect of Raymond's widely influential essay by an otherwise perceptive commentator, Ko Kuwabara (2000):

> …The Cathedral and the Bazaar, is a metaphorical reference to two fundamentally different styles of software engineering. On the one hand, common in commercial development, is the Cathedral model, characterized by centralized planning enforced from the top and implemented by specialized project teams around structured schedules. Efficiency is the motto of the Cathedral. It is a sober picture of rational organization under linear management, of a tireless watchmaker fitting gears and pins one by one as he has for years and years. On the other hand is the Bazaar model of the Linux project, with its decentralized development driven by the whims of volunteer hackers and little else. In contrast to the serene isolation of the cathedral from the outside, the bazaar is the clamour itself. Anyone is welcome – the more people, the louder they clamour, the better it is. It is a community by the people and for the people, a community for all to share and nurture. It also appears chaotic and unstructured, a community where no one alone is effectively in charge of the community. Not all are heard or noticed, and not all are bound to enjoy the excitement. For others, however, the bazaar continues to bubble with life and opportunity.

But "the bazaar" remains a peculiar metaphor for a system of production: the stalls of actual bazaars typically are retail outlets, passive channels of distribution rather than agencies with direct responsibility for the assortment of commodities that others have made available for them to sell. Given the extensive discussion of the virtues and deficiencies of "the bazaar" metaphor that was stimulated by Raymond's (1998a) essay, it is all the more remarkable that what has managed to pass with scarcely any comment is rhetorical finesse of the problem of aligning the activities of producers with the wants of the needs and wants of the final, non-specialist users of these information-goods.[11]

In contrast, the tasks set in our project on free and open source ('Libre') software represent an explicit response to the challenge of providing *non-metaphorical* answers to the classic economic questions of whether and how this instance of a decentralized decision resource allocation process could achieve coherent and socially efficient outcomes. What makes this an especially interesting problem, of course, is the possibility of assessing the extent to which institutions of the kind that have emerged in the free software and open source movements are enabling them to accomplish that outcome – without help either from the "invisible hand" of the market mechanism driven by price signals, or the "visible hands" of centralized managerial hierarchies.[12] Meeting this challenge requires that the analysis be directed ultimately towards providing a means of assessing the social optimality properties of

[11] See, e.g., Kuwabara (2000), and references in the notes accompanying Raymond (1999: pp.19-63): "Cathedrals and Bazaars."

[12] Benkler (2002) has formulated this problem as one that appears in the organizational space between the hierarchically managed firm and the decentralized competitive market, focuses attention primarily on the efficiency of software project organizations, rather than considering the regime as a whole.

the organization and management of "open science", "open source" and kindred cooperative knowledge-creating communities. In all such circumstances were specialized expertise of those participating is critically important for the effective conduct of the work, and prior evaluations are made difficult the asymmetric distribution of the pertinent bodies of expert knowledge and knowledge about expertise, one would expect to find a greater reliance upon ex post verification and validation of the work-product, rather than on a formal management tools for selecting the producers and monitoring the quality or intensity of their contributions. When considering the issues surrounding the nature and efficacy of the coordination, governance and quality regulating mechanisms that have emerged in the context of large and complex Libre software projects, it is therefore relevant to recognize the potential tensions between the product control devices that can be readily implemented by expert-developers and those which may be of importance to the end-users in the society at large.

2. Modelling Libre communities at work

The parallels that exist between the phenomena of "open source" and "open science," to which reference already has been made, suggests a modeling approach that builds on the generic features of non-market social interaction mechanisms. These involve feedbacks from the cumulative results of individual actions, and thereby are capable of achieving substantial coordination and coherence in the collective performance of the ensemble of distributed agents. This approach points in particular to the potential significance of the actors' consciousness of being "embedded" in peer reference groups, and therefore to the to role of collegiate recognition and reputational status considerations as a source of systematic influence directing individual efforts of discovery and invention.

Our agent-based modeling framework has been structured with a view to its suitability for subsequent refinement and use in integrating and assessing the significance of empirical findings about patterns of resource allocation within large and more complex F/LOSS projects, well known exemplars of which are the Linux operating system, the Mozilla web-browser and the Apache web-server. Systematic empirical evidence about the participants in such projects, their behaviors, patterns of communication and the internal modes of project organization has only lately begun to be collected.[13] Nonetheless, to guide initial specifications it is possible to draw upon insights provided by experienced project leaders and descriptive generalizations about micro-level incentives from survey- and interview-based studies, regarding the nature of the community norms that might not only affect the mobilization of participants, but guide the allocation of software developers' efforts within particular projects. Nothing in that approach invites hypothesizing the operation in the representative, 'ideal-type' F/LOSS community of a system of social norms that mimics the particular features of collegiate reputational reward systems such as are found in the Republic of Science, but the postulation that an equivalent functional structure exists is an entirely

[13] Pioneering studies of large projects include the work of S. Koch and G. Schneider (2000) ;Tuomi (2000, 2001); Dempsey et al. (1999, 2002); S. Krishnamurthy (2002). More recent studies have sought to exploit new methods of automated data-mining from source code repositories, and to build links between that information and data on communications flows among project participants. On patterns of authorship and the structure of code within large projects, obtained using the CODD data extraction algorithm (developed by R. A. Ghosh and V. V. Prakash (and described first to measuring the code size of projects in the Orbiten Free Survey (2000) [see http:www.orbiten.org/codd]), see Ghosh (2003)]; for findings from the application of CODD to studies of sequential releases of the the Linux kernel see Ghosh and David (2003). See Gonzalez-Barahona and Robles (2003, 2004); Robles, Koch and Gonzalez-Barahona (2004); Gonzalez-Baharona, Lopez and Robles (2004).

plausible basis upon which to proceed. Further, it is equally clear that provision eventually will need to be made to incorporate functional equivalents of the conventions and institutions governing recognized claims to scientific 'priority' (being first), as well as the symbolic and other practices that signify peer approbation of exemplary individual performance.

A systems analysis perspective such as is familiar in general equilibrium economics suggests that within such a framework we should be capable also of asking how the norms and signals available to micro-level decision-takers in the population of potential participants will shape the distribution of resources among different concurrent projects, and direct the attention of individual and groups to successive new projects. That, in turn, will affect the growth and distribution of programmer's experiences with the code of particular projects, as well as the capabilities of those who have gained familiarity with the norms and institutions (e.g., software licensing practices), and the coordination and communication styles specific to individual projects, as well as the more widely shared practices of the Libre software regime. Obviously, the formation of generic knowledge and capabilities provides potential "spill-overs" to other areas of endeavor – including the production of software goods and services by commercial suppliers. From this it follows that to fully understand the dynamics of the Libre software mode and its interactions with the rest of the information-technology sector, one cannot treat the expertise of the software development community as a given and exogenously determined resource.

From the foregoing it should be evident that the task upon which we are embarked is no trivial undertaking, and that to bring it to completion we must hope that others can be drawn into contributing to this effort. We report here on initial progress towards that goal: the formulation of a highly stylized dynamic model of decentralized, micro-level decisions that shape the allocation of Libre software programming resources among project tasks, and across distinct projects, thereby generating an evolving array of Libre software system products, each with its associated qualitative attributes. In such work, it is hardly possible to 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, because it is only reasonable to suppose that these may influence how they allocate their personal efforts in this sphere. At this stage, it is not necessary to go into great detail on this matter, but among the many motives enumerated it is relevant to separate out those involving what might be described as "independent user-implemented innovation."[14] Indeed, this term may well apply to the great mass of identifiably discrete projects, because a major consideration 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.[15] 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 force 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. "Independent" is the

[14] 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."

[15] Just how great a mass of these independent projects represent in the total remains unclear, as the most readily available indications are those obtained by studying the characteristics of the just the *publicly announced* open source projects. On the basis of gathered data from Sourceforge.net on the 100 most active projects observed in the "mature stage" (i.e., the final stage of a project's development, when it is almost fully functional and distributed), Krishnamurthy (2002) reports finding that the modal project has only 1 identified developer; among the most active projects –a mere fraction of the 40 thousand-odd listed on that site -- the median number of developers was 4.

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 Libre software community." The individuals engaging in this sort of software development may use open source tools and regard themselves as belonging in every way to the free software and open source movements. Nevertheless, it is significant that the question of whether or not their products are to be contributed to the corpus of non-proprietary software, rather than being copyright-protected for purposes of commercial exploitation really is one that they need not address *ex ante*. Being essentially isolated from active collaboration in production, the issue of the disposition of authorship rights can be deferred until the code is written.[16] That is an option which typically is not available for projects that contemplate enlisting the contributions of numerous developers, and for which there are compelling reasons to announce a licensing policy at the outset.

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 Libre software production in "community-mode" or, for convenience *C-mode,* contrasting it with software production in *I-mode* (Dalle & David, 2003). 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, at this point, to support interesting model-building. Consequently, our attention here focuses exclusively upon creating a suitable model to simulate the actions and outcomes of populations of Libre software 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 Libre software developers' therefore does not presuppose that career considerations of "ability signaling," "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.[17]

---

[16] In this respect it can be argued that the decision of the individual developer working in *I-mode* to participate in Libre software production actually is not a decision about the mode of production, but, instead is a matter of making and *ex post* choice of whether or not to disclose the source code, and whether or not it is worth trying to exploit the resulting program as protected intellectual property. The economics of such post-production decisions certainly are of interest, and the normative force of the open source and free software movements may come into play at this stage. The represents a promising line for future research, but it is a line of inquiry quite different from the one we are pursuing here.

[17] It will be seen that the probablistic allocational "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 allocational rules.

Our model-building activity aims to provide more specific insights not only into the workings of Libre software communities, but also into their interaction with organizations engaged in proprietary and "closed mode" software production. It seeks to articulate the interdependences among distinct sub-components of the resource allocation system, and to absorb and integrate empirical findings about micro-level mobilization and allocation of individual developer efforts both among projects, and within projects. Stochastic simulation of such social interaction systems is a powerful tool for identifying critical structural relationships and parameters that affect the emergent properties of the macro system. Among the latter properties, the global performance of the Libre software mode in matching the functional distribution and characteristics of the software systems produced to the evolving needs of users in the economy at large, obviously is an issue of importance for our analysis to tackle.

It is our expectation that in this way it will be feasible to analyze some among the problematic tensions that may arise been the performance of a mode of production guided primarily by the internal value systems of the participating producers, and that of a system in which the reward structure is tightly coupled by managerial direction to external signals deriving from the satisfaction of end-users' wants. Where the producers are the end-users, of course, the scope for conflicts of that kind will be greatly circumscribed, as enthusiasts for of "user-directed innovation" have pointed out.[18] But, the latter solution is likely to serve the goal of customization only by sacrificing some of the efficiencies that derive from producer specialization and division of labor. The analysis developed in this paper is intended to permit investigations of this classic "trade-off" in the sphere of software production.

3. The model[19]

3.1 Structure and rationale

The core of the stochastic simulation model of open source software production presented here is a behavioral kernel: heterogeneous developers face an existing set of software modules[20] – about the state of which we assume that they are fully informed[21] –, and they choose the module they will contribute to stochastically, according to their effort endowments and to the reward that each module can grant them. Heterogeneity, represented here by the existence of a stochastic (discrete) choice function, classically accounts for all the un-observed characteristics of each developer. Each developer will prefer to undertake the

[18] See von Hippel (2001), Franke and von Hippel (2002), on the development of "user toolkits for innovation," which are specific to a given production system and product or service type, but, within those constraints, enable producers to transfer user need-related aspects of product or service design to the users themselves.

[19] The current version of this model, and its exposition, have enormously benefited from various comments and criticisms we have received from various people after we had previously opted for an "early" release (Dalle & David, 2003), precisely to elicit comments both from the academic community and also from participant observers in open-source projects. Any modelling exercise like this one implies some conscious level of abstraction and simplification: however, the modellers might not be immediately accurate in their modelling attempts, over-estimating some parameters while underestimating others, and therefore critically need insights and inputs from many other experts. Needless to say, this basic assumption still completely holds here.

[20] Which would probably correspond more to packages than to individual files according to the terminology in vigour in most open-source projects.

[21] Which implies that each new contribution is immediately made accessible to all developers. We do not account for now for the fact that some contributions are suitable not to be integrated in the code, depending notably on their relevance, and on maintenance policy, at least at the module level: see section 3 below for simulation experiments with various global maintenance rules.

most rewarding tasks, according to a reward system still to be determined: however, this is not a deterministic choice as there are necessarily unobserved heterogeneous characteristics which drive this choice, and for which no model can account for if it wants to avoid to absolute contingency trap in which it would fall if it assumed it could take all relevant variables into account. A simple, and now relatively traditional way to handle this (Anderson, de Palma & Thisse, 1992), is to consider that the more rewarding modules will be chosen with a higher probability – or, in the statistical physicist's language now common in most disciplines including economics, to consider rewards as weights and to compute the probability that each module is chose according to a ratio between its weight and the sum of all weights, possibly distorted by various parameters and coefficients. Namely:

$$\mathbf{P}\left[\textit{chosen module} = (\textit{virtual})\ \textit{module } m\right] = \frac{\rho_m(\alpha)}{\sum_{\textit{all modules}} \rho_i(\alpha) + \sum_{\textit{all virtual modules}} \rho_{i'}(\alpha)} \qquad (2.1)$$

Where $\rho_{\square}(\alpha)$ stands for the reward of α contributed to each module.

Among un-observed characteristics, an important caveat concerns here the precise nature of the problems that each developer faces in its own idiosyncratic situation, a feature which is reportedly known as a significant determinant of developer choice among various open-source projects: from Eric S. Raymond's "Every good work of software starts by scratching a developer's personal itch." (Raymond, 1998a) to Eric von Hippel's user theory (Harhoff, Henkel & von Hippel, 2000; Franke & von Hippel, 2002; von Hippel, 2002) and to more recent and quantitative evaluations (see e.g. Ghosh, Glott, Kreiger & Robles, 2002; David, Waterman & Arora, 2003; Lakhani, Wolf, Bates & DiBona, 2003). This is something that we only deal with stochastically in the current version of the model: namely, we account for un-observed characteristics like this one, but we do not specify it fully yet. A later version of this model should involve the development of such an improved behavioral kernel, which would account for the matching process between developer and module characteristics – not underestimating the precautions that would be needed to support the validity of the claims then obtained as these developments would increase the non-ergodicity (David, 2001), and perhaps the deterministic features, of the system, due to a higher number of variables which would then accounted for.

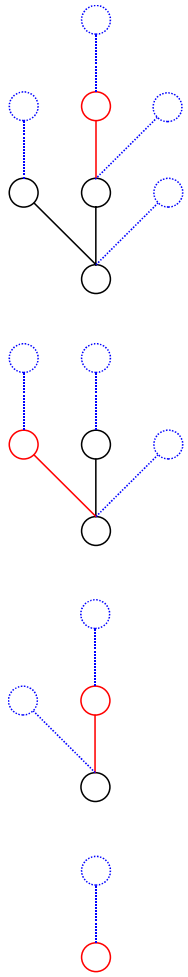

Figure 1

Graphical representation of a software system growth process as an upward evolving tree

It would mean developing a new module. Indeed, this is exactly what open-source software development generally implies, since open-source developers do not simply consider adding their efforts to existing modules, but they also create new ones to supplement existing ones, when appropriate: this is precisely the mechanism that we have implemented to induce simulated code growth. To do so, we consider the following modeling finesse: we suppose that to each existing module is associated a 'virtual' module, which stands for the eventuality that a new module could be created from the existing one, either by developing an existing functionality out of it, in the form of an external module, or simply by adding a new one which would supplement this module: clearly then, the new module and the existing one would be technically linked[22], the new external module would typically be included in the existing during the compilation process, or sometimes simply called. Figure 1 represents the growth of a software system according to this rule: at each step, red lines and circles represented the last created module, while blues lines and circles represent virtual modules attached to each existing one, and black lines and circles represent older modules created during earlier steps.

[22] Also in the sense of the wording of the GPL licence, for instance, which implies that if the "parent" module was GPL'd, then the new one would also be.

In this framework, the emerging architecture of the modules is indeed mathematically a tree, since, by construction, there are no loops and each module is linked to only one (parent) module. This tree does not completely correspond to the actual directory tree, nor to the full set of technical and functional dependencies, which are usually known as the architecture of a software system per se (Bass, Clements & Kazman, 1998), since some of technical dependencies are not accounted for here, namely the fact that some modules can be called by several others. We have rather characterized it here as an emerging architecture, i.e. the one which stems from the fact that developers generally decide to create a new module to solve a technical problem they face while working on a particular existing one, or as a development or part of an existing module. Therefore, this emerging architecture here has much to do with the kind of phenomenon that Herbert A. Simon (1962) famously characterized years ago in a seminal article on the "architecture of complexity", and we indeed feel very much intellectually indebted to him, all the more so as the emerging architecture that he considers is also a tree-like "hierarchical system"[23]: Simon indeed precisely suggested that the emerging architecture of complex systems tended to often be spontaneously such, *because complex systems were born out of simple ones, and because simple systems then tend to be somehow included in more complex ones*. As for our modeling of open-source software development, the rationale for the emergence of a hierarchy of modules is strongly similar: a complex system is dynamically born out of a simple one; new modules are created out of existing ones to supplement them by developing existing functionalities or adding new ones; and these new modules can be included in higher ones during the compilation process or at least are called as sub-systems. We are also very close here to the recent research on modularity (Baldwin & Clark, 2000), and extending the model further in this direction, notably by studying more extensively, and modeling more accurately, the actual technical interactions between modules, would also be a very fruitful research avenue

This model then allows us to test one of the main hypotheses that have been suggested about software development in open-source mode, namely, what we suggest to call the "regard" hypothesis. According to this theory, developers are significantly influenced by reputation effects: Eric S. Raymond (1998ab) was among the first to emphasize this idea in the famous essays he wrote as a participant observer in open-source communities; and it has been since suggested repeatedly by several other important studies of open-source software development, also as a more general attempt to analyze the striking similarities between open-source and open science communities (Benkler, 2001; Kelty, 2001; Dalle, David & Steinmuller, 2002)[24]. In a companion paper to this one (Dalle, David, Ghosh & Wolak, 2004), we indeed suggest that open-source software falls into a broader category which we characterize as peer regard economies: not reputation in a traditional sense, but rather to account for the fact that in these economies the actions undertaken by developers should *at the margin* account for the relative regard of their peers about their deeds[25].

We therefore suggest that developer statistically tend to prefer lower-level modules to higher-level ones in the hierarchical structure presented above, since the former, more general ones, are regarded as more generally relevant by their peers than more specialized ones, and also because their visibility being higher, it will automatically grant their contributors more regard from their peers. Contributing to the Linux kernel is deemed a potentially more rewarding

[23] But not in the traditional sense of hierarchy, just as a description of an architecture with several levels: indeed, so as to avoid mis-understanding, the French translation of this paper has precisely selected a word meaning "tree-like" (*arborescent*) to translate "hierarchy".

[24] On the economics of Open Science, see Dasgupta & David (1987, 1994), David (1998abc, 2000).

[25] At least when they are in C-mode, as opposed to I-mode: see Dalle & David (2003).

activity than contributing to the file system, and the latter still dominates writing an obscure driver for a newly-marketed printer. Stated differently, we postulate here that there is a strong dependency between the emerging hierarchical architecture of the software system and the associated hierarchy of peer regard. Yet in other words, we postulate that there is lexicographic ordering of rewards based upon a discrete, mainly technically-based "tree-like" structure formed by the successive addition of modules. Clearly, this is an important assumption that should be tested empirically: in this respect, our companion paper presents preliminary elements in this direction, by showing that the pattern of signed and un-signed contributions in the Linux kernel is not random, and tends to show that technical dependencies tend to play a relatively significant role, among other factors (Dalle, David, Ghosh & Wolak, 2004).

Still according to the "regard" hypothesis, and to also account for other observations by Raymond and others, we also add the two following properties influencing developer choice:

[a] Launching a new project is more rewarding than contributing to an existing one, all the more so when several contributions have already been made: namely, the first contributions to a given module are more rewarding than later ones – this is more or less analogous to the "first to publish" rule in open science communities, and it seems to be also relevant in open source ones.

[b] Contributing to an active project is more rewarding that contributing to a stagnant or dormant one, as contributions will simply be more noticed by a larger number of peers.

This last property could be considered as a second-order effect, since it supposes that developers and contributions will be attracted by modules that already have drawn more numerous contributions, inasmuch as developers share take signals from one another's behavior as to which modules are "interesting." But it also is a relevant consideration for individuals seeking peer regard that one's contributions, however technically astute, should have an audience.[26]

[26] The empirical study by Dalle, David, Ghosh and Wolak (2004) of the proportions of unsigned (uncredited) code in modules of the Linux kernel finds that the number of developers contributing to the package exerts no appreciable *independent* effect on the probability that code is signed (credited). This result – and other related findings – are understandable, as Dalle et al. point out, if it is borne in mind that signing one's code is likely to be a more important means of gaining recognition (and approbation) for developers who make comparatively smaller contributions to multiple projects, and who join projects in their later, more mature growth phases, that it is for major core developers who become identified with a project during its early stages by virtue of their extensive contributions to its technically critical modules. Were it to be 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 size of the relevant audience – i.e., in the total number of developers engaged in contributing to the module in question. Yet, the findings on the joint determination of average code-signing propensities and developer participation in the modules of the Linux kernel does not support such a supposition of exogeneity. Rather, it appears that larger modules (measured in terms of code size) exert a selective drawing power that results in larger average contributions of code per developer. If considerations of peer regard underlay the bias in the selectivity effect on contributors, that would suggest not only that the number of developers contributing to a given module was an endogenous variable, but that the selectivity effect (itself was an indirect reflection of considerations of peer regard on the part of early contributors of larger blocks of code) worked to vitiate the emergence of a positive statistical association between the proportion of code that was signed and the total number of developers in the module.

3.2. Mathematical description

In mathematical terms, we therefore get:

$$\forall m \text{ a module: } \rho_m(\alpha) = r_m(x_m + \alpha) - r_m(x_m) \qquad (2.2)$$

Where $\rho_m(\alpha)$ still stands for the expected[27] reward of contributing α to module m, $r_m(\cdot)$ is the cumulative reward function, i.e. the total reward associated with the sum of all contributions to module m, x_m is the current improvement of module m, i.e. precisely the sum of all past contributions, and α is a potential contribution for a developer's given effort endowment. Clearly then, by construction, for m' the virtual module associated with m:

$$\forall m' : x_{m'} = 0 = r_{m'}(x_{m'}) \qquad (2.3)$$

Thus:

$$\forall m' \text{ the virtual module associated with module } m : \rho_{m'}(\alpha) = r_{m'}(\alpha) \qquad (2.4)$$

And $r_{\cdot}(\cdot)$ is a (positive) increasing convex function in coherence with rule [a] above, which imposes that the first contributions are more rewarded than the later ones.

We will further consider here that:

$$r_m(x_m) = r_{d(m)}(x_m) = v_{d(m)}(x_m)\, d(m)^{-\lambda} \left(\left(1 + c(m)\right)^{\gamma} \right) \qquad (2.5)$$

Where $d(m)$ is the distance of module m from the first "root" module; $v_{d(m)}(x_m)$ is the function which gives the version number of module m at distance $d(m)$ from the root from its current improvement x_m; $c(m)$ is the number of contributions received by module m, and $\lambda \geq 0$ and $\gamma \geq 0$ are parameters.

This simplification of $r_m(\cdot)$ into $r_{d(m)}(\cdot)$ is a direct consequence of the hierarchical and lexicographic assumption presented above: the reward associated with module m depends on its location in the software architecture only as it depends from the height of the module in the hierarchical module tree, $d(m)$. This dependency is then given according to characteristic exponent λ: when $\lambda = 0$ all modules are similarly rewarded, whatever their height $d(m)$, while as λ goes to infinity the dependency of rewards to the height of the module increases, with:

$$r_0(x_m) = v_0(x_m) \left(\left(1 + c(m)\right)^{\gamma} \right) \text{ and } \forall m \neq root : r_m(x_m) \to 0 \text{ as } \lambda \to +\infty \qquad (2.6)$$

Since, by construction, the height of a virtual module is the height of its parent plus one, (2.4) and (2.5) above imply that:

[27] Part of the reward at least, especially for new modules, depends upon other contributions to be added later: therefore its expected nature.

$$\forall m : \rho_{m'}(\alpha) = r_{m'}(\alpha) = r_{d(m)+1}(\alpha) = v_{d(m)+1}(\alpha)\left(d(m)+1\right)^{-\lambda} \qquad (2.7)$$

If of course we note also that :

$$\forall m : \left(\left(1+c(m')\right)^{\gamma}\right) = 1 \text{ since } \forall m : c(m') = 0 \qquad (2.8)$$

By construction: the term in $c(m)$ in equation (2.5) above allows us to account for rule [b], namely, to render the more active projects – the "hot spots" – more rewarding for further contributions – even more and more so as γ increases, while this effect disappears completely when $\gamma = 0$. It is therefore not relevant for potential virtual modules, and the mathematical expression has been chosen in consequence.

We then define also:

$$v_{d(m)}(x_m) = \log\left(1 + x_m d^{\mu}\right), \qquad (2.9)$$

where $\mu \geq 0$ is another characteristic exponent, which simply implies that it is easier to increase version numbers for high modules than for lower ones, and we easily verify that $v_{d(m)}(x_m)$, and therefore $r_{\square}(x_m)$ are positive increasing convex functions of x_m.

To complete the description of the model, what we are finally missing is a distribution of effort endowments α within the population of independent developers[28], normalized by individual productivities to directly translate into potential improvements added to modules: that is, a distribution of the size of contributions[29]. On the basis of the relative sizes of the high- and low-activity segments of the developer population found by various surveys, and notably the FLOSS survey, we suppose that these endowments are distributed according to an exponential distribution function[30]. Using the classical inverse transformation method on the cumulative distribution (e.g. Ross, 2003), we then compute the following exponential random number generator, which generates contributions α from a uniformly distributed probability:

$$\alpha = -\frac{1}{\delta}\ln(1-p), \qquad (2.10)$$

[28] Whatever their unit of measurement, typically in SLOC or in KLOC: if such a measure was to be selected, it should be noted that we do not differentiate here between lines added, replaced and deleted. As a consequence, a more appropriate measure of improvement would then be the sum of all lines added, replaced, and deleted.

[29] Since, as we mentioned above, this model is for now a model of contributions and not a model of contributors: the heterogeneity of contributions is a consequence of the heterogeneity of contributors, and we do not track for now for individual developers and for instance for the history of their contributions, which would necessarily imply to attach idiosyncratic characteristics to each individual developer. As a consequence, the model presented here is not properly speaking agent-based, but is more stochastic in its nature, accounting better for the intrinsic heterogeneity of economic actors through the observable heterogeneity of their actions.

[30] For now, we do not make any distinction different types of contributions, be they patches to correct bugs, or the addition of new features – which Raymond (1998a) indeed characterizes as the correction of "bugs of omission". This aspect of the model could certainly also be improved in later versions. We do not account either for the involvement of commercial developers: we have started doing experiments in this respect, which will be reported elsewhere.

where $p \in [0;1]$ is uniformly distributed and δ is a parameter which controls for the mean of the distribution, as a straightforward calculation will show that $\langle \alpha \rangle = \frac{1}{\delta}$, where $\langle . \rangle$ stands for the means.

Simulation experiments can then be run easily according to this model, in discrete time: at each time step a new contribution is simply added to the existing system[31], i.e. either an existing module is improved or a new one is created. The procedure is the following:

i. A random contribution is given by (2.10) ;

ii. The rewards of all existing modules, considering their current improvements, and of all virtual modules are computed according to (2.5), (2.7), and (2.9);

iii. A module is chosen according to (2.1), and the system is then modified in consequence.

Figure 2 represents the typical growth path of such a system, and should therefore be compared to Figure 1 above (numbers for each module are version numbers).

To finish with the mathematical description of the model, we just need to add that our ultimate goal is to analyze some of the characteristics of the emerging software systems, described here as code trees: in particular, we are interested in measure how sensitive their morphology (software-tree forms) is to parameter variation. Just to push the tree metaphor further, the obvious trade-offs of interest are those between intensive effort being allocated to the elaboration of a few "leaves," i.e. modules, which may be supposed to be highly reliable and fully elaborated software systems whose functions in each case are nonetheless quite specific, and the formation of a "dense canopy" containing a number and diversity of "leaves" that, typically, will be less fully developed. Indeed, a simple way to characterize this morphology, which we will use below, is simply to compute the Gini coefficient of the distributions of the sizes all "leaves" – modules.

The reason for this is that an important empirical finding, reported by Ghosh & David (2003), is that the Gini coefficients of the distribution of module sizes tend to be very (indeed, extremely) high. As for now, these results were obtained for the Linux kernel. This relatively striking feature means that there is a very limited number of modules with receive numerous contributions, and a very large number of modules with only a limited number of contributions, maybe only one[32].

[31] As in all the experiments presented here, starting only with the root module with initial improvement 1.

[32] We would suggest that one-contribution (and therefore one-contributor) modules realized in I-mode can eventually be contributed to the project according to a global C-mode behavior (Dalle & David, 2003).

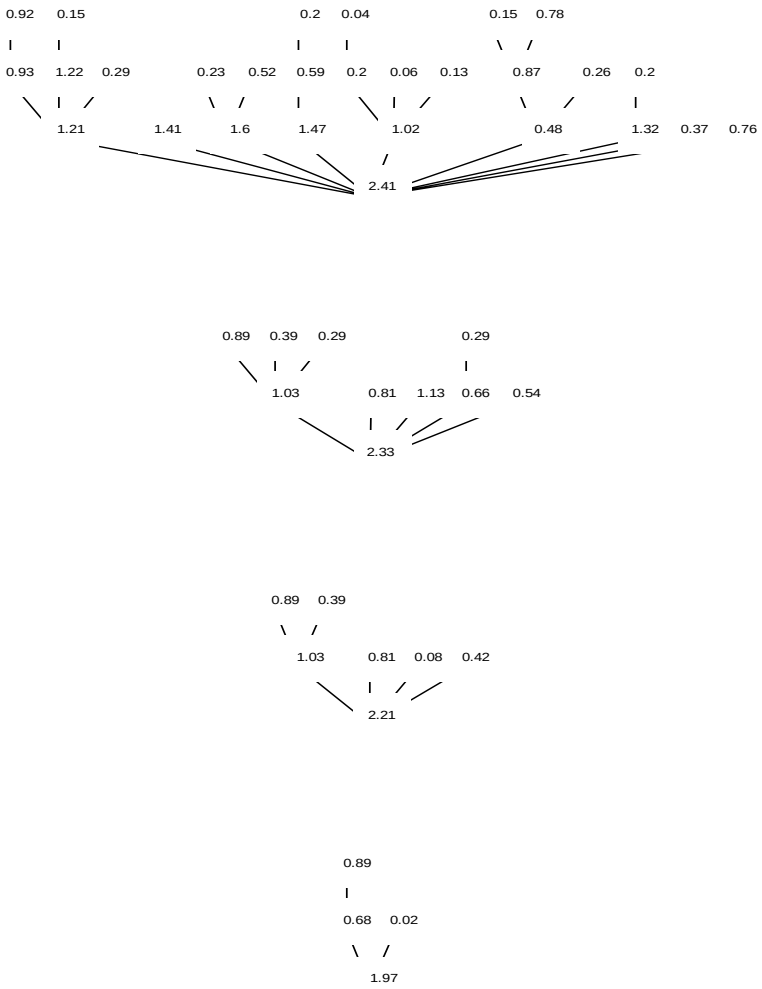

Figure 2

A simulation of the growth of a software project

But we also believe that the emerging morphologies of such software systems are absolutely non-neutral with regard to their social utility. Clearly, this should be and must be here a highly debated issue, but the reason why we dare enter this area is because we really feel critical to make some progress in the understanding of this difficult problem: for the purposes of this first step, the focus of the analysis is confined to showing the ways in which the specific norms of the reward system and organizational rules can shape emergent properties of software systems, such as its range of functions and reliability. Indeed, the global performance of development in open source mode, in matching the functional and other characteristics of the variety of software systems that are produced with the needs of users in various sectors of the economy and polity, obviously, is a matter of considerable importance that will bear upon the long-term viability and growth of this mode of organizing production and distribution.

Therefore, we introduce here a simple social utility function, which basically captures 3 principles, which we first make clear:

(1) Lower modules are more socially valuable than higher ones because more users use them, and because also of the range of other modules and applications that eventually can be built upon them;

(2) A greater diversity of functionalities is more valuable because it provides software solutions to fit a wider array of user needs;

(3) Users value greater reliability, which is likely to increase as more work is done on the code, leading to a higher number of releases. Releases that carry higher version numbers are likely to be regarded as 'better' in this respect.

We then capture these ideas together according to the following[33] social utility function:

$$u = \sum_{m}^{(modules)} \left[\left(1 + v_d(m) \right)^{\nu} - 1 \right] d^{-\xi} \qquad (2.11)$$

Where $\nu \in [0;1]$ and $\xi \geq 0$ are parameters, again in the form of characteristic exponents: obviously, ν controls rule (3) above, while ξ controls rule (1) and while the summation in itself accounts for rule (2).

4. Simulating the allocation of efforts in open-source software development

For the sake of the exposition, Figure 3 (in annex) presents a typical collection of trees generated in the context of the simulation experiments presented in this section. In any case, all simulations reported in this paper have been conducted with similar values of the parameters, excluding λ and γ since they control the main regard effects that we intend to test, and which we will therefore keep as parameters, namely:

$$\begin{cases} \delta = 3 \\ \mu = 0.5 \\ \nu = 0.5 \\ \xi = 2 \end{cases}.$$

Which, at this point, can be considered as reasonable numerical values, all the more so similar results to the ones presented here hold for other values in the same range, except for higher values of ξ which tend to eliminate the existence of non-corner maxima to social utility, by typically, and logically, driving the maxima toward low very high values of λ.

4.1 Simulation results on project architecture an the distribution of module sizes

Table 1 and Figure 4 now present Gini coefficients measuring the degree of concentration of the module-size distribution for various values of λ and γ, i.e. depending on the strength of the two main "regard" effects in the model: λ controlling the influence of the inner hierarchy of modules within the project, and γ controlling the attractivity of "hot spots" – active modules. Clearly, there is a region of the parameter-space within which both coefficients exert a positive influence on the Gini coefficient: one can see the boundary of that region describes a steeply rising "ridge-line" in Table 1 along which the entries for G attain a maximum in the neighborhood 0.86-.88. The row-maxima and column maxima for the Gini coefficient, which coincide along that ridge-line are marked in boldface in the table. In other words, there is a linear combination of λ and γ that constitutes a limit, above which the

[33] In the future, we might be willing to implement a better differentiation between functionality and reliability, with the idea also that different users might typically value both aspects differently.

software system fails to develop, so that virtually all the code growth is confined to a single (root) module.[34]

We certainly do not generate Gini coefficients as high as those found in actual open-source project code (sometimes over 0.99), but this would have been impossible due to the limitations of our stylized simulation experiments; furthermore, we do not account for the technical peculiarities of some specific modules in a software project like Linux – where the modules providing a great variety of "drivers" results in a multiplicity of comparatively small code-packages, which contributes the projects high Gini coefficient. But simulations that displayed even the results did not hold in various simulation experiments that we conducted, and that are not described here in detail, for which Gini coefficients typically remained low (i.e., rarely exceeding 0.5). This was for instance the case when we grew software systems:

i. Without rule [b] above, i.e. without the "hot spot" effect;

ii. Without rule [b], but with another rule, [c], accounting for a negative effect that a higher number of existing modules stemming from any given one would have on the motivation to create still another child to this module.

We would therefore suggest, according to these results but also to the other ones presented below which similarly exhibit high Gini coefficients, that there is a positive correlation between the existence of regard-based reward structures, and specially fashion effects, and the observed characteristics of package size distributions within some open-source software projects.

4.2 Simulation results on developers' choices, project "release" rules, and social utility

To turn now to results about social utility, Table 2 and Figure 5 show that social utility varies systematically with λ and γ. But the effect of higher γ, raising the attractiveness of "hot spots" of developer activity among the modules, is to monotonically reduce the social utility of the overall project code. In Table 2 only the column maxima are marked in boldface, to highlight the fact that these occur at successively lower values of λ as the attractiveness of "hot spots" is increased, and that the value of the column maxima themselves decreases. It will be seen, therefore, that the locus of column maxima, and therefore the maxima of *social utility nowhere correspond* to the ridge-line region of Gini coefficients that appears in Table 1 and Figure 4 (Note that the γ axis has been inverted between the Figures 4 and 5, in order to obtain greater clarity in the perspective imposed by the 3-D view).

Although these results are remain quite tentative, it is difficult to escape the conclusion that the 'regard' motivations which we have hypothesized to operate within the open-source software communities of large projects are not conducive to generating socially optimal, or even second-best optimality in the emerging functional design of software systems. To put it differently, and still more hypothetically, if the motivations of independent developers drive them to take decentralized decisions that are responsive to "peer regard" and imitative of "social fashion" within the project-community (which would correspond to specifying parameters λ and γ in the "high Gini" zone), then the results could be considered as a less socially beneficial global outcome, compared to other situations were fashion and regard effects would typically have less potency in guiding developer's decisions.

[34] A close approximation to this boundary line is found as: *max-Gini* = (0.1) λ + (0.43) γ. As one may see, this relationship begins to break down for value of $\lambda < 1$.

		γ										
		0,0	0,2	0,4	0,6	0,8	1,0	1,2	1,4	1,6	1,8	2,0
	0,0	0,47	0,47	0,47	0,48	0,47	0,48	0,50	0,55	0,61	0,70	0,68
	0,5	0,48	0,47	0,47	0,47	0,49	0,50	0,54	0,60	0,70	0,83	**0,73**
	1,0	0,48	0,48	0,48	0,49	0,51	0,53	0,58	0,67	0,82	**0,87**	0,67
	1,5	0,48	0,49	0,50	0,52	0,53	0,57	0,65	0,75	0,85	0,72	0,41
	2,0	0,50	0,50	0,51	0,54	0,58	0,61	0,74	0,86	**0,87**	0,71	0,32
	2,5	0,50	0,52	0,53	0,57	0,61	0,69	0,79	**0,88**	0,81	0,43	0,30
	3,0	0,52	0,53	0,56	0,60	0,65	0,74	0,85	0,87	0,61	0,38	0,17
	3,5	0,53	0,56	0,59	0,63	0,70	0,79	**0,88**	0,72	0,58	0,15	0,17
	4,0	0,55	0,57	0,61	0,66	0,75	0,84	0,86	0,61	0,56	0,22	0,05
	4,5	0,57	0,60	0,65	0,70	0,79	**0,87**	0,83	0,52	0,22	0,05	0,00
λ	5,0	0,59	0,62	0,67	0,74	0,82	**0,87**	0,79	0,46	0,25	0,05	0,00

Table 1: Gini coefficient for module size distribution

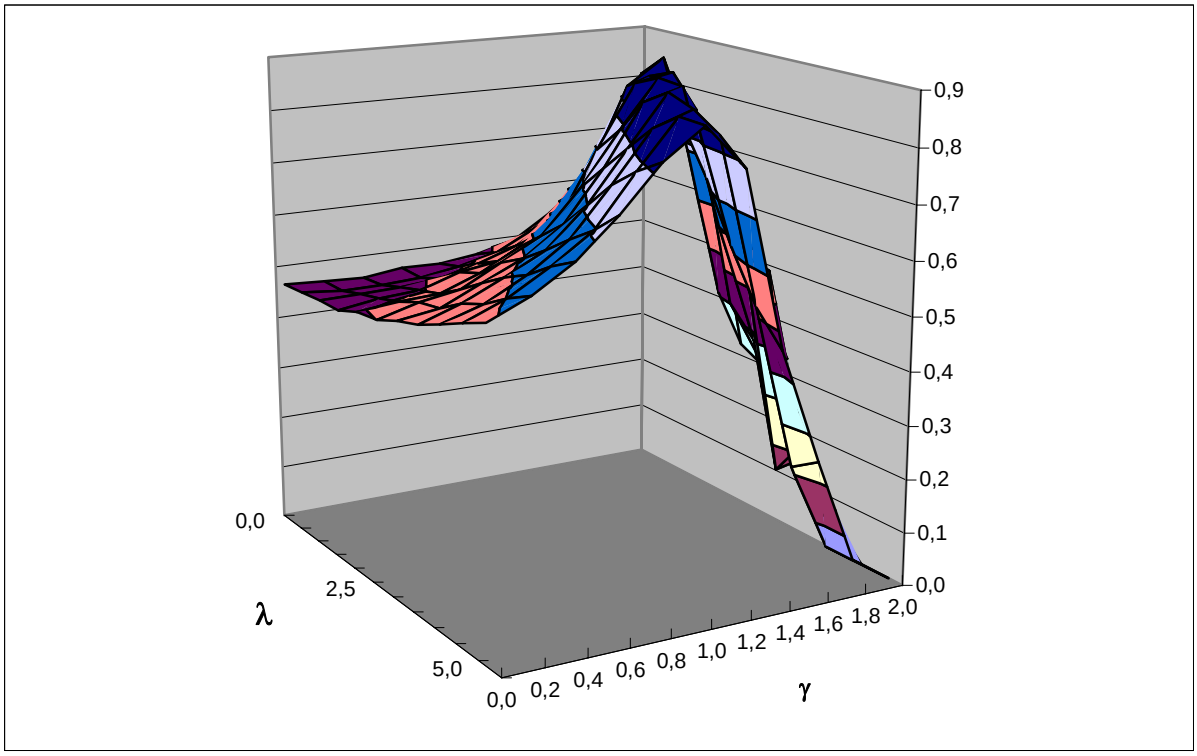

Figure 4: Gini coefficient for module size distribution

Needless to say, this rather striking conclusion rests entirely on the specification of the social welfare criterion, as well as the other behavioral specifications of the model. That it overturns the results reported by Dalle and David (2003) on the basis of an earlier version of the model is not problematic in itself: the present model, as has been seen, incorporated a previously overlooked "externality effect" – in the form of mimetic behaviors affecting individual choices about what parts of the project to which code is contributed, and effect that "social fashion – that enables the model to capture an important empirical feature of large projects' code, namely, the skewed distribution of the module sizes. But, there are still other empirical regularities, such the characteristics of the distribution of individual developer contributions to each of the modules, that the model in its still present, highly simplified form cannot simulate. Therefore, it is undoubtedly premature to attach any finality and certainly any policy significance to the finding just reported.

Nevertheless, in view of the importance and intrinsic theoretical interest of understanding the factors that will affect the assessment of open-source project performance from the viewpoint of external evaluators, and final users in particular –which our social welfare function seeks to represent, we believe it is appropriate to call attention to the foregoing results. At very least, it exposes the "instability" of the results yielded by the model during this still early phase of the sequential modification of its specifications. Indeed, one can do no less than report such reversals in results, if we are to adhere to the general scientific norm of "full disclosure" – placing one's trust in the latter's efficacy in promoting rapid, cumulative advances in knowledge.

		γ										
		0	0,2	0,4	0,6	0,8	1	1,2	1,4	1,6	1,8	2
	0	5,1	5,2	5,0	5,1	4,7	5,1	5,2	5,0	4,6	**4,3**	**2,4**
	0,5	7,1	6,3	7,0	6,8	6,7	6,4	7,0	5,6	**5,4**	3,5	2,3
	1,0	8,4	8,4	7,8	8,6	8,2	8,2	7,2	**6,5**	4,3	3,0	2,0
	1,5	10,0	9,5	10,1	9,5	9,4	8,8	**8,0**	6,2	3,4	2,1	1,7
	2,0	11,2	11,4	11,0	10,5	10,1	**9,2**	6,9	4,3	2,6	1,9	1,7
	2,5	12,3	12,1	11,4	11,2	10,2	8,6	6,4	3,2	2,2	1,7	1,7
	3.0	13,3	13,2	12,3	11,4	**10,5**	8,0	4,8	2,8	1,9	1,7	1,6
	3,5	13,8	13,4	12,4	**11,9**	9,6	7,0	3,7	2,0	1,8	1,6	1,6
	4.0	14,4	13,8	**12,6**	11,5	8,8	5,7	2,7	1,8	1,7	1,7	1,6
	4,5	14,6	**14,1**	12,4	10,6	7,8	4,6	2,4	1,8	1,6	1,6	1,6
λ	5.0	**15,0**	13,8	12,2	9,7	6,9	3,4	2,2	1,7	1,6	1,6	1,6

Table 2: Social utility

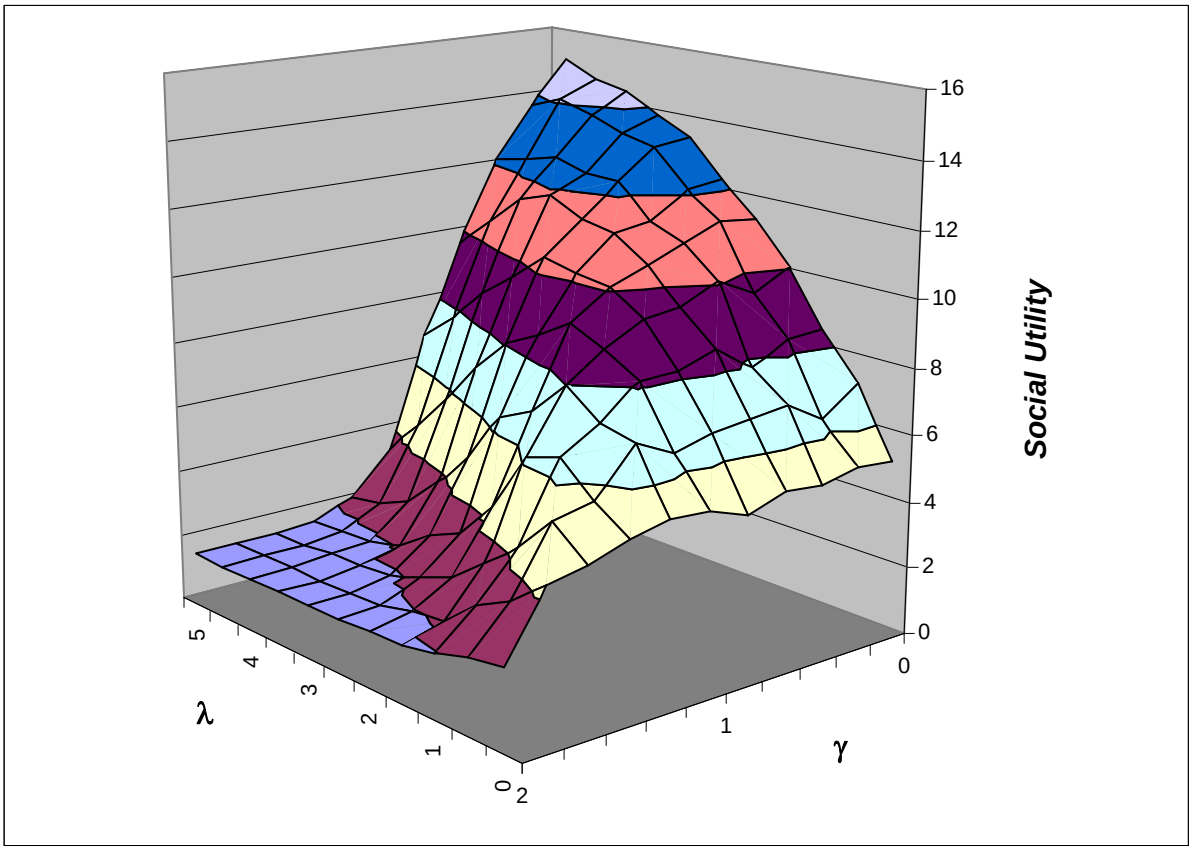

Figure 5: Social utility (without maintainers)

5. Conclusion

We have reported in this paper of an effort to construct a simulation model of software development in Libre (open-source) mode. Obviously, the next steps can be taken in either of two directions. Following the empirical path and the iterative development, we can seek to calibrate the model more precisely by using the empirical regularities (e.g., on the types and sizes of modules, and the overall architectural morphology) observed in a number of large open-source projects of various kinds. But there are also clearly are a number of research agenda items in view on the analytical path, many having been set out by our first report on this undertaking (Dalle and David 2003), with several new ones being added in the course of the foregoing discussion. Perhaps the most important discrete elaboration will be the steps from modeling the tree to modeling a forest: adding typically a second “project tree” that may compete with the first for developers’ contributions but also benefit from experience that they gain in working on the “rival” project. Next we envisage exploring whether the dynamics of the system becomes markedly more complex when the forest expands to beyond two trees, allowing some projects to have relationships marked by complementarity whereas other pairs are substitutes as far as the production relationships are concerned.

Looking ahead on both paths, it seemed obvious that it will be beyond our power to adequately pursue on our own even the principal items in the vast research agenda that we

have opened – at least not at a rate that can keep up with the proliferating sources of empirical data that a fully specified model could illuminate, and the multiplying policy questions that a carefully parameterized model could be used to analyze. Consequently, in a somewhat self-referential fashion, we are moving towards facilitating the conduct of the simulation project in the distributed open-source manner. The next version (release) of the model will provide not only the mathematical structure of a modularized version of the simulation structure, but the source code we are running, and which others may use to replicate our results and modify the structure.

Whether this should formally become an experiment in the organizing of this kind of research on open-source as an open-source-like project (with all that this implies about claims to copyrights, licensing terms and governance norms), is an intriguing question. But, for the present, the "open science" mode seems to be powerful and attractively familiar way in which to move forward, inviting others to join in the collective advancement of knowledge.

REFERENCES

Anderson, de Palma and Jacques-François Thisse. 1992. "Discrete Choice Theory of Product Differentiation." Cambridge, Mass.: MIT Press.

Baldwin, Carliss Y. and Kim B. Clark. 2000. Design Rules: Vol.1. "The Power of Modularity." Cambridge, Mass: MIT Press.

Bass, Len J., P. Clements and Rick Kazman. 1998. "Software Architecture in Practice". Addison-Wesley.

Benkler, Yochai. 2002. "Coase's Penguin, or Linux and the Nature of the Firm." Yale Law Journal. 112: 369-446.

Bonaccorsi, Andrea and Cristina Rossi, 2003. "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 <http://www.osdn.com/bcg>

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.

Dalle, Jean-Michel. 1997. "Heterogeneity vs. externalities: a tale of possible technological landscapes", Journal of Evolutionary Economics 7: 395-413.

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, revised version submitted to First Monday.

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

Dalle, Jean-Michel, Paul A. David, Rishab Aiyer Ghosh and W. E. Steinmueller,

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

Dalle, Jean.-Michel, P. A. David, Rishab A. Ghosh, and W. E. Steinmueller, 2004 "Advancing Economic Research on the Free and Open Source Software Mode of Production," (Publication forthcoming in *Building Our Digital Future:Future Economic, Social & Cultural Scenarios Based On Open Standards*, Marleen Wynants and Jan Cornelis, Eds., Brussels: Vrjie Universiteit Brussels (VUB) Press, 2005.) [Pre-print as SIEPR Discussion Paper (December 2004) available at: http://siepr.stanford.edu/programs/OpenSoftware_David/NSFOSF_Publications.html.]

Dalle, Jean-Michel, Paul A. David, Rishab Aiyer Ghosh and Frank Wolak. 2004. "Free and Open Source Software Developers and the Economy of 'Regard': A Quantitative Analysis of Code-Signing Patterns within the Linux Kernel," paper presented to the EPIP3 Seminar, Schuola Superiore Sant'Anna, Pisa, April, and to the Oxford Workshop on Libre Software (OWLS), Oxford Internet Institute, June.

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.

David, Paul A. 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. 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.

_____ 2002a. "La coopération, la créativité et al cl^ture des débats dans les sciences, in Institutions et innovation: De la recherche aux systèms sociaux d'innovation." (Sous la direction de Jean-Phillipe Touffut) Paris:Bibliothèque Albin Michel Economié, pp. 67-104.

_____ 2002b. "Cooperation, Creativity and Closure in Scientific Research Networks: Modeling the Simpler Dynamics of Invisible Colleges," SIEPR/CEEG-Social Science and Technology Seminar Series Paper (December 4, 2002). [Available at: http://siepr.stanford.edu/programs/SST_Seminars/David_All.pdf].

_____ 2003. "The Economic Logic of 'Open Science' and the Balance between Private Property Rights and the Public Domain in Scientific Data and Information: A Primer," in National Research Council, *The Role of the Public Domain in Scientific Data and Information*, Washington, D.C.: National Academy Press/

_____ 2004. "Understanding the Emergence of Open Science Institutions: Functionalist Economics in Historical Context," *Industrial and Corporate Change*, 13(1), August.

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.]

Dempsey, B., Weiss, D., Jones, P. and Greenberg, J. 1999. "A quantitative profile of a community of open source Linux developers." SILS Tech. Rep. TR-1999-05, School of Information and Library Science, University of North Carolina at Chapel Hill, (October). [Available at: www.metalab.unc.edu/osrt/.

_____2002. "Who is an open source software developer?" *Communications of the ACM,* Volume 45, Number 2 (2002), Pages 67-72.

Elliott, Margaret. S. and Walt Scacchi. 2003. "Free Software Development: A Case Study of Software Development in a Virtual Organizational Culture." April. http://opensource.mit.edu/papers/eliottscacchi.pdf

Feller, Joe and Brian Fitzgerald. 2002. "Understanding Open Source Software Development." Addison-Wesley: UK.

Franke, Nikolaus and Eric von Hippel. 2003. "Satisfying heterogeneous user needs via innovation toolkits: the case of Apache security software." Research Policy, 32 (7): 1199-1215, Special Issue on open source software development edited by Georg von Krogh and Eric von Hippel.

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

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, 15th February). http://opensource.mit.edu/papers/ [Forthcoming in Joe Feller, Brian Fitzgerald, Scott Hissam, Karim Lakhani, eds., *Making Sense of the Bazaar,* Cambridge, MA: MIT Press, 2005]

Ghosh, Rishab Aiyer and Ved Prakash, Vipul. 2000. "The Orbiten Free Software Survey," *First Monday*, 5:7. http://www.firstmonday.org/issues/issue5_7/ghosh/

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). http://opensource.mit.edu/papers/

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/.

Gonzalez-Barahona, Jesus and Gregorio Robles. 2003. "Free Software Engineering: A Field to Explore," *Upgrade*, IV(4), August..

Gonzalez-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. http://opensource.mit.edu/papers/

Gonzalez-Barahona, Jesus and Gregorio Robles. 2004. "Getting the Global Picture," A presentation at the Oxford Workshop on Libre Software (OWLS), Oxford Internet Institute, 25-26 June 2004. [Available at: http://www.oii.ox.ac.uk/fiveowlsgohoot/postevent/Barahona&Robles_OWLS-slides.pdf].

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* 6 (12: 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.

Lakhani, Karim and Eric von Hippel. 2000. “How Open Source Software Works: "Free" User-to-User Assistance.” Research Policy 32 (6): 923-943.

Lakhani, Karim R., Bob Wolf, Jeff Bates and Chris DiBona, 2003. “The Boston Consulting Group Hacker Survey (in cooperation with OSDN)”. [Available at: <http://www.osdn.com/bcg/bcg-0.73/BCGHackerSurveyv0-73.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.

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.

Ramil Juan F. & Neil Smith. 2004. “Qualitative simulation of models of software evolution”, Journal of Software Process – Improvement and Practice, forthcoming.

Raymond, Eric S. 1998a. "The Cathedral and the Bazaar," *First Monday*, 3 (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). [Available at: http://firstmonday.org/issues/issue3_10/raymond/index.html and www.tuxedo.org/~esr/writings/homesteading.

Robles, Gregorio, Stefan Koch and Jesus Gonzalez-Barahona. 2004. “Remote Analysis and Measurement by Means of the CVSAnalY Tool,” Working Paper, Informatics Department, Universidad Rey Juan Carlos, (June). [Available at: http://opensource.mit.edu/papers/robles-koch-barahona_cvsanaly.pdf.]

Ross, Sheldon M. 2003. “Introduction to Probability Models.” 8th Edition. Academic Press.

Simon, Herbert A. 1962. “The Architecture of Complexity.” Proceedings of the American Philosophical Society, 106 (December): 467-482.

Tuomi, Ilka. 2000. “Learning from Linux: Internet Innovation and the New Economy,” Working Paper (February) [available at: http://www.jrc.es/~tuomiil/articles/LearningFromLinux.pdf.]

_____ 2001.“Internet, Innovation, and Open Source: Actors in the Network,” *First Monday* 6(1), Jan. 8, 2001.

von Krogh, Georg, Sebastian Spaeth and Karim R. Lakhani. 2003. “Community, joining, and specialization in open source software innovation: a case study.” Research Policy, 32(7): 1217-1241

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.