

Micro-dynamics of Free and Open Source Software Development

Lurking, laboring and launching new projects on *SourceForge*

By

Paul A. David

Stanford University & Oxford Internet Institute
pad@stanford.edu

Francesco Rullani

Sant'Anna School of Advanced Studies & Copenhagen Business School
fr.ivs@cbs.dk

Version 5

December, 2006

Submitted for publication in *INDUSTRIAL AND CORPORATE CHANGE*

ACKNOWLEDGEMENTS

We are grateful to the staff of *SF.net* staff for providing a copy of the *SF.net* archive, and to Paola Giuri, Salvatore Torrisi, Matteo Ploner, Gaia Rocchetti for providing the “cleaned” SFnetDataset that was created by their project (with Rullani’s participation). Interviews with *SF.net* developers were of a significant help in interpreting the data from the archive. Research for this paper has had the benefit of discussions with Laurent Daudet, Giacomo Pasini, Alessio Moneta, Alessandro Sapio, and Randolph Bruno. In addition this version has been improved by other comments on earlier presentations, made at the OWLS-Paris Workshop (April, 2006), the OSSWatch Conference on Sustainability and Open Source Software in Oxford (April, 2006) the OSS2006 conference in Como (June, 2006), and the International Schumpeter Society Conference in Nice (June, 2006). David acknowledges the financial support of the National Science Foundation (NSF Award IIS 0326529) and the Stanford Institute for Economic Policy Research’s KNIIP Program. Rullani gratefully acknowledges the financial support his work received from the CE, DGXII, FP V Project “Economic Change: the micro foundations of institutional and organisational change: *EconChange*”; and from the Laboratory of Economics and Management of the Sant’Anna School of Advanced Studies in Pisa, Italy.

Abstract

Quantitative methods are employed to describe two fundamental processes in the creation of free (libre) and open source software (FLOSS) that are at work in the collaborative development environment of the *SourceForge.Net* platform: *resource mobilization* and “*entrepreneurial initiatives*” which generate new development projects. The micro dynamics of the individuals’ involvements in these processes are analysed by defining “activity states” that correspond to “lurking” (not contributing or contributing to projects without become a member), “laboring” (joining one or more projects as members), and “launching” (founding one or more projects). The transition probability matrices constructed from observations on the activities of 222,835 individuals who registered on *SF.net* (during a 14-month period, mainly in 2001) characterize first-order Markov chains describing processes that are ergodic. The computation of the limiting “equilibrium” distribution of individual joining and launching activities is not used here to produce long-run predictions, because the time window of the available data is too short. Instead, it is conceived as an instrument to isolate the main forces acting in the underlying entrepreneurial and recruitment dynamics at work on the platform “shaking off” the weaker tendencies. It is shown that, although only a small proportion of the considered cohorts of *SF.net* registrants become even minimally active, the active “core” of project members and project founders is able to attract an increasing number of developers. *SourceForge* is seen to be more than an attractor of projects that are being “born again” under open source licenses: this virtual collaborative development environment shares the regenerative properties of tangible “industrial districts” that give rise to new, innovative enterprises. Implications for the *exploitation* and *exploitation* processes at work in the FLOSS model and about its *sustainability* are also derived.

Keywords: open source software, collaborative development environments, industrial districts, project founding, project joining, entrepreneurship and social communication skills, SourceForge, Markov chain models

1. Introduction

When a Free/Libre/Open Source Software (FLOSS¹) project releases its code under an open source license, this enables developers who have not worked on that software to examine it, modify it, discover the source of its performance defects, and repair them, and (under some conditions that depend upon the nature of the license²) redistribute both the original and the versions that carry their modifications. Other developers may then enter the production process, cooperating with the project founder and its members to further enlarge and enhance the reliability and functionality of the code basis. The cooperative process is open and “fluid”: non-members of the project may contribute by submitting bug reports, patches and requests for particular features; project members are free to leave, just as new developers can be added at each moment to the project’s “group” or “community”. Every developer decides how and how much to contribute. When authoritative structures of governance emerge, they typically formalize and render explicit informal processes that were already at work in the network of collaborators (Mateos Garcia and Steinmueller, 2003). The multiplicity of projects, each having its own organization history, generates a variety of “experiments” in different governance structures.

Similarly decentralized mechanisms are at work in the relationships among distinct projects. Precisely because the development process is undertaken on an open and voluntary basis, coordination among projects is not assured. On the supply-side, projects may overlap in some of the features they provide for users and developers.³ On the demand side, projects may compete for resources -- i.e. developers with specific competences or simply developers who are willing to report bugs, or able to submit “patches” that fix a problem. Moreover, new projects may be launched at every moment by anyone, and may then attract other developers, forming a community around them; or they may remain individual undertakings that fulfill the specific needs of their founders. They may “fork” – splitting into multiple teams who develop variants of the original software, or they may simply be abandoned and enter a moribund state without attaining a so-called “mature” status, in which activity maintaining the code continues sporadically at low levels. These individual and collective organizations co-evolve, competing for and sharing resources (both expertise and re-usable code), as well as visibility, commercial sponsors of their development and distribution activities, final users and developers, originating a series of different structures interlaced by changeable links. In their competitive and cooperative interactions, these open source development projects form ecologies.

1.1. The virtual collaboration environment

One such ecology is observable on the SourceForge platform (at <http://sourceforge.net>), referred to hereinafter as *SF.net*. This collaborative development environment is not unique, as *FreshMeat* and *Savannah* being other notable sites hosting open source software projects, but it is far and away the most densely populated in numbers of project groups and members. In that respect the object of our study constitutes an important microcosm of the FLOSS universe, in which a great variety of organizations are found to co-exist, some closely resembling one another, and others being very different in size, organizations structure, and purpose. That is not the same thing as saying that it is a statistically representative portion of the FLOSS universe. Indeed, plainly it is not, because *SF.net* does not host the very largest and emblematic projects engaged in what Dalle and David (2005) label “C-mode” (for community-mode) production of FLOSS, many of which, like the Linux kernel, Apache, and Mozilla, maintain their own sites. Nonetheless, 73 project groups that were in the membership size range from 30 to 102 on 10th January 2003, the date of

¹ A discussion of the FLOSS movement’s characteristics and of its impact on software production can be found in Raymond (1998a, 1998b). To have an idea of the research on this topic see Lerner and Tirole (2004), Maurer and Scotchmer (2006) and Dalle et al. (2005).

² On the distinctions between different Open Source licenses see, among others, Lerner and Tirole (2005) and Giuri *et al.* (2002).

³ Consider for example one of the most relevant projects on *SF.net*: *phpMyAdmin* (awarded with the “2006 Community Choice Awards”). The homepage of the project describes it as “phpMyAdmin is a tool written in PHP intended to handle the administration of MySQL over the Web. Currently it can create and drop databases, create/drop/alter tables, delete/edit/add fields, execute any SQL statement, manage keys on fields.” But other *SF.net* projects have very similar features: *remote mysql manager* is described as “PHP code to manage a MySQL db over the web, Add/Drop databases and tables; Insert/Update/Delete records. Enter your own SQL query. No config files/scripts. Small and simple.” Similar duplicative software offerings can be found among other categories of projects hosted by *SF.net*.

the last observations used in this study, including BZFlag, Freenet, Python, and JBoss⁴, share the same kinds of coordination, contributor recruitment and governance challenges that distinguish *C*-mode from the very small projects that are led and staffed by one or two individuals (*I*-mode). The mass of *SF.net* projects, if one simply counts projects not weighted by any performance measure or code size or membership, are situated at the latter, extreme lower end of the size spectrum: in the dataset we study, of all projects that had at least one member, 67 percent had no more than one member.

1.2. Virtual and actual “industrial districts”

The question of whether the particular environment upon which this study focuses is statistically “representative” is less germane for our purposes than the opportunities it affords for studying the patterns of behavior over time of a very large number of individuals in relation to a correspondingly large number of projects that co-exist within a defined “locale” – particularly a locale that facilitates information flows and has a potential for inducing collaborative interactions. Viewed from the economic geographer’s perspective, some salient features of the virtual “space” created by *SF.net* bear a striking resemblance to key processes observed in the regional, physically bounded “industrial districts” and agglomerative clusters of “high-tech” enterprises. In the virtual space of the platform, as in those Silicon Valley-like spaces where “industrial localization” has occurred (to use Alfred Marshall’s term), there is a continuous movement of agents from the state of “potential entrants” to more active states, where many become engaged in already existing organizations (firms), and some start new ventures. Like software projects, firms can be successful and grow in terms of numbers of employees, or fail, so that entrepreneurs can decide to abandon their enterprises and re-enter an inactive state, can apply to be hired by other existing firms, or start a new firm from scratch. Moreover, as in real districts, platforms such as *SF.net* hold a particular attraction for individuals and groups interested in (FLOSS) project development because they offer facilities that reduce costs of searching for and recruiting “workers” (projects members) as well as for “managers” (project administrators and maintainers). In addition, emulating the positive externalities of co-located social interactions, the virtual collective collaboration environment provides lower set-up costs for internal communications via email subscriber lists and project forums; similarly it affords “entrepreneurs” (projects’ founders) ready means of advertising the purposes of their project and affords potential contributors and users alike comparatively easy access to the source code from which both the present state of performance of a project’s software and the pace of growth and improvement may be assessed. The open, cooperative practices of FLOSS communities in disclosing the process of producing and patching code quite obviously facilitate the circulation of information about software system architectures and sub-routines, just as open access to project descriptions of internal organizational structures and practices facilitates the circulation of members and knowledge of managerial and governance practices.⁵

For the purposes of the analysis that follows, the aspects of resemblance that seem most salient are those derived from the fact that, like the classic industrial districts, the ensemble of FLOSS community activities gathered on *SF.net* are able to benefit from the agglomeration economies and innovation opportunities created by the exploitation of recombinant novelties. These arise from the gathering of a critical mass of actors with diverse capabilities, and a variety of projects within a circumscribed (virtual or physical) space; and from the provision on the platform of “infrastructural” facilities that emulate the cost-reducing effects that industrial producers enjoy as a result of their collective ability to induce not only a large pool of suitably skilled workers, but the complementary clustering of highly specialized agents supplying inputs, brokerage, insurance and financial intermediation adapted to the requirements of the firms localized in the region. From the latter come Marshall’s economies of scale that are external to the firms but internal to the district; whereas it is from the circulation of agents and the openness of the code that emerges the Marshallian “atmosphere” in which “the secrets of industry are in the air.”⁶ Indeed something more than a

⁴ Notice that some projects host some features outside *SF.net*. For example Python (at <http://www.python.org/>) and JBoss (at <http://www.jboss.org/>).

⁵ As a rule, such information about the internal workings of business firms in places like Silicon Valley remains much more difficult for outsiders to obtain, even by individuals who are prepared to spend many lunch-times in personal conversations canvassing the opportunities of moving from company or partnership to another in the same locale.

⁶ On this distinction, and the dynamical implications of Marshallian factor input market externalities, see, e.g., David and Rosenbloom (1990), and David, Foray and Dalle (1998).

metaphorical pairing of these districts with the open source communities' way of working can be seen in the role of information circulation and in the positive externalities and their feedbacks into the innovation process. Consider this summary passage in Lécuyer's (2005) richly documented and insightful account of "the making of Silicon Valley" in the era before the 1970's:

"The constant circulation of design, production, and management skills within the district also explains the "success" of individual corporations and the manufacturing cluster as a whole. Manufacturing processes, design methodologies, and managerial techniques moved quickly from firm to firm and from industry to industry....Skills and techniques flowed within industries as engineers moved from one corporation to the next....This circulation [of practices and techniques] was partially predicated on the culture of cooperation that electronics hobbyists developed on the San Francisco Peninsula. Radio amateurs valued collaboration and the sharing of information. Litton, a radio ham and microwave engineer, freely shared his knowledge of vacuum techniques and tube production methods. He also helped other entrepreneurs start electronics firms in the area....Also critical were the dense networks of personal relationships that emerged in the region....Because they raided each other's employees and benefited from informal contacts among their engineer, MOS startups on the Peninsula developed a repertoire of process "tricks" which were known only in the area. These tricks enabled MOS companies to introduce new products rapidly to the market and to obtain good manufacturing yields."

The parallels between the open flows of information that characterize open source software development communities, and the circulation of information within Silicon Valley in its formative era, and among the firms in the innovative Italian industrial districts,⁷ interesting as they may be, will not be pursued further on this occasion. Rather, we use the metaphoric conceptualization of *SF.net* as a virtual industrial district to propose a practical empirical approach to studying the sustainability of FLOSS production and distribution communities as a distinct sector of the global software industry.

As is true of an actual industrial district, sustainability and the capacity for growth depend upon the ability of the locale to mobilize resources and innovative capabilities to which it potentially has access. Thus, the virtual district has to be able to attract developers to enter the site and be drawn into active participation (i.e. to move from a passive "lurking" state into the progressively more active states). The questions to be answered become: To what extent do movements occur from states of "participation" (characterized by sporadic contacts with one or more projects) to a state featuring more committed involvement as members of a project? How typical are such "project joining" moves in the experience of the mass of individuals that have entered the *SF.net* environment? Do developers who are project members virtually from the outset tend to go on to join other projects, and do they do so with higher or lower frequencies than those who are non-members at the same phase of their career on the platform? What steps lead developers into the role of project founding, and how likely is it that an individual who enters this environment without a current attachment to a project will launch a new one? Having founded one project, what is the expected duration until the individual launches another?

1.3. Resource mobilization, innovation and sustainability: the organization of the study

The first part of this study, which is set out in sections 2-4, focuses on a systematic statistical analysis of the project-joining and project-founding activities of individuals observed on *SF.net* over a period of two years. Our empirical model and the statistical methodology used to estimate it are explicitly dynamic, and because they do not impose preconceptions about one or another particular sequence of movements among roles in the FLOSS community, this approach contrasts with previous studies that have described differentiated roles and assumed (or presupposed) that individuals must migrate step-wise from peripheral and infrequent actions toward roles that are successively more and more central for the production of a project's code. The most widely known schema for this structure is the "onion model" (Crowston and Howison, 2005), which represents agents in the stylized FLOSS production community as being positioned in layers of bug fixers, bug reporters, occasional participants in communications with a project's email

⁷ A study by Maggioni (2004) has drawn explicit parallels between the Italian industrial district and the FLOSS community "model of production", based on stylized descriptions of the two. Like the passage quoted from Lécuyer (2005), this is quite suggestive of points of similarities in production methods and managerial policies.

subscriber list, each layer being the farther and farther away from a core of developers who are regularly contributing to the code basis of the project. Much of the research devoted to measuring the relative numbers of the community members occupying these differentiated roles has taken the form of careful and quite detailed case studies of specific projects. But the early implementations of this approach have been static, and their analyses of cross-section data for a single project contrasts with a dynamical framework that allows for the possibility that developers might concurrently occupy the same role in more than a single project, or could take different roles in several projects.

A small number of previous studies, however, have been concerned explicitly with the sequential dynamics of FLOSS “project joining.” The pioneer research on this subject was carried out by von Krogh, Spaeth and Lakhani (2003) in a case study of Freenet, a project developing a peer-to-peer system (which as was noted above is among the biggest projects hosted on *SF.net* in the period of analysis). By examining the chronological record of individuals’ appearances in email subscriber lists and forum records of bug reports and patch submissions, as well as of the tasks performed as members of the project, the authors identify an implicit “joining script” defined by the type, frequency and intensity of individual’s engagement; they then measure the distribution of durations that were required for contributor to make the passage from the periphery to the core of the project’s code development activities. Jensen and Scacchi (2005) also have studied processes of “recruitment and role migration,” focusing on the transitions from the role of end-user toward involvement in code development in a number of projects that were selected as case studies. But, in contrast to the notion that there is “a” joining script, they find a variety of paths, some of which do not conform at all well to the conventional conceptualization of step-by-step progressions from periphery toward the core.

In a still more recent case study of GNOME (a well known large project developing components of a FLOSS desktop environment, which enjoys sponsorship from several companies through its foundation <http://foundation.gnome.org>), Herraiz *et al.* (2006) also ask whether the conventional dynamic interpretation of the onion model is a good representation of the joining process, and seek answers by estimating the distribution of transit times between developers’ first instances of performing characteristic tasks in the onion’s successive layers. The authors report that although it is typical for individuals in the GNOME community to have contacted the project by email before beginning to commit code⁸, conventional suppositions regarding the next step in the sequence are violated by most of the GNOME developers’ behaviors: for example, they often commit to the project’s CVS before sending their first bug report. Moreover, this study finds striking evidence that there are important heterogeneities in the developer population in the directness and speed with which individuals enter core development work.

Rather than undertaking to add further to the case study evidence, the empirical strategy adopted here exploits the *SFnetDataset* (i.e. the dataset at our disposal, as defined more precisely in the next section) by analyzing the transitions made by individuals in the population on the platform between pairs of “activity states” that distinguish more crudely among the variety of roles identified in the “onion model”. This approach uses the chronological ordering of observations on individual’s activities to estimate the frequencies (defined for a uniform specified time interval) with which non-members make contact with projects after having been completely passive, or return to a spell of passive lurking following an interval of activity in submitting bug reports, patches or feature requests. Similarly, to study project joining, it is possible to measure the frequencies with which active individuals move from non-member status into membership of one of the projects on the platform; or emerge as project founders.

The transition frequencies must be defined carefully with respect to standard intervals of time, ordered by reference to a common point in the life of the individuals on the platform. When arranged in a matrix, the maximum likelihood estimates of the state dependent probabilities of transitions define Markov chains that describe the sequence of the platform resident’s movements among the regions of “the onion” – not confined within a single project, but allowing for transitions across projects and even into core development roles as founders of new projects.

In section 2 we describe the nature of the data obtained from *SF.net*, the definition of the state space for the Markov model of transitions among distinct roles/activities, and the procedure followed in

⁸ Every time a developer changes the code basis of a project, CVS commits are produced. Thus, their number gives an idea of developers’ rate of activity in the production of code and of their role into the project.

establishing the uniform time constants to be used to measure individual transitions (see Appendix 1 for details of the time series analysis). Section 3 presents the estimated transition probability matrix describing the dynamics of “project joining and founding” and examines its properties. Section 4 discusses the use of the ergodic property of the estimated transition matrix to describe the limiting (or equilibrium) distribution towards which the distribution of the population on the platform is driven when it shakes free of the influence of the initial activity states in which agents are observed.

The study’s second part, in sections 5 and 6, focuses on the activities of the “entrepreneurial core” that emerges among the ranks of those who have been drawn into this environment. The Silicon Valley experience and the recent history of Italian North-East regions show that the “entrepreneurial spirits” that the district produces and attracts, and which continues to flourish there, constitutes a fundamental factor in the explanation of the spectacular industrial development and economic growth experienced by these regions (Saxenian, 1994; Becattini, 2001). The process often entails spin-offs and reapplication, including re-mixing of knowledge and capabilities acquired in co-located enterprises, but when new firms are launched, new ideas are introduced and novel and existing energies are mobilized along new trajectories. In the FLOSS context, where “entrepreneurs” can be identified as those developers who launch new projects, these agents play an analogous parallel role in expanding the existing code basis and in furnishing the whole FLOSS community with new ideas to implement and with new trajectories to develop. The capacity of FLOSS development environments to generate new ventures, i.e., to induce or to support project-launching, is no less important for the long term sustainability of this sector of the software industry than the recruitment of project contributors. Developers working in already existing projects represent the *exploitation* side of the *SF.net* community, whereas “founders” represent the *exploration* side of its production model.

Notice that nothing is said here about the eventual outcomes of project founding activity in this virtual district: our analysis is confined to examining the strength of project founding *propensities*, not their success or failure to reach maturity with a fully developed code basis. Nevertheless, the industrial district analogy suggests that one should expect only a small proportion of new ventures to survive competition from diverse projects that already are established and attracting developers, as well as competition for users from projects with very similar goals. The process of selective survival renders the system as a whole *dissipative*, i.e. it is based on a mechanism “burning” a high amount of resources to produce a comparatively small number of surviving innovations.

Section 5 examines the differences between founders and project members with respect to an array of fixed characteristics, including the activities in which they were observed at the very beginning of their time on *SF.net*, and prior to the periods to which the following dynamic analysis relates. Then, having defined who “entrepreneurs” are, their activity is described over time by means of a set of Markov chains. The first group of estimated Markov chains is obtained by stratifying the population into subsamples according to developers’ prior activity status. A second stratification according to the timing of the foundation activity provides a second group of transition matrixes. In both cases the relative transition matrixes lead to specific limiting distributions of developers among the activity-states which can therefore be compared to the initial ones. Section 6 performs a similar exercise employing expected “career histories,” i.e. the number of cumulated projects founded and joined by each developer over the observed period, particularly for those individuals who become founders, and alternative interpretations of the findings are discussed. Section 7 comments on the broad significance of our findings for the question of the sustainability of the FLOSS system of production, and concludes the paper by briefly considering ways in which future research could transcend the limitations of this exploratory implementation of the Markovian approach to analyzing the population dynamics of ecologies of FLOSS developers and their projects.

2. Data and Methods of Analysis

SF.net is an on-line platform created by VA Software (former VA Linux), where FLOSS developers can meet and coherently organize their activities on different projects. Following the foundation of *SF.net* late in 1999, VA Software collected and organized a data archive containing a chronological record of the activities of developers and projects hosted on the platform. The statistical analysis in this paper is based upon a portion of that archive, an edited dataset – referred to here as the *SFnetDataset* – that covers the whole *SF.net* population of developers (222,835 individuals) who registered on *SF.net* during the 14 months from September 1, 2000 through October 26, 2001. There are other datasets relative to *SF.net*. For example, the FlossMole project (former OssMole project, see <http://ossmole.sourceforge.net/> and Howison *et al.*,

2005) gathers together and gives coherency to different data sources (mainly obtained from platforms as *SF.net* or *Freshmeat*) describing projects and developers' activities. Another example is the recent initiative undertaken by a group of researchers of the Notre Dame University, who have released under a specific license a series of monthly dumps of the *SF.net* data archives (see <http://www.nd.edu/~oss/Data/data.html>).

Several previous studies have drawn data from the *SF.net* site and its repositories to answer a variety of questions concerning the phenomena of open source software producers and production. The following list is not presented as inclusive, but suggests the richness and multi-dimensional nature of this information source. Krishnamurthy (2002), in a pioneering ecological study, examined a sample of project groups from the platform, using several indicators to select 100 "mature" projects and finding that few among them actually were associated with development "communities", whereas the great majority had but a single group member and at best were only sporadically active in email discussions and commits to their respective CVS repositories. More recently, Madey *et al.* (2004) have mapped the network of developers' collaborations as members of project groups, Fershtman and Gandal (2004) sought to identify the determinants of variations in projects' average output of code per contributor-member, Lerner and Tirole (2005) examined the effects of variant forms of the open source license on project growth, and Comino *et al.* (2005) identify factors determining projects' attained stages of development. Still more recently, Giuri *et al.* (2006) have exploited information from the *SFnetDataset* (the same archival data as is used here) in order to study the diversity of skills, experience and the division of labor among the members of *SF.Net* projects extant during 2000-2002; while Robles and Gonzalez-Barahona (2006) have utilized time stamp and other information collected on first registration to discover the geographical distribution of the more than 1,180,000 individuals who were listed as registered on *SF.net* in November 2005.

2.1. The *SFnetDataset*

As is the case with many valuable deposits, there are non-negligible "refining" costs entailed in exploiting the material extracted from *SF.net*. Some errors have been found in the procedures followed by *SF.net* maintainers in building summary statistics and storing the data (see Hunt and Johnson, 2002). Moreover, even when the collected numbers are corrected, instances of developers' improper use of the facilities provided by *SF.net* somewhat degrade the accuracy of the resulting statistics (see Howison *et al.*, 2005). To prepare the *SFnetDataset* on which this study is based, the original material from the archive was first "cleaned" by application of a variety of *ad hoc* procedures, and a systematic algorithm for inferring founders' identities from partial information has been implemented.⁹

A number of previous studies have pointed to a different class of "perils and pitfalls" of working with *SF.net*. Most cautions that have been issued along these lines relate not to errors in the data itself, but to mistakes in analysis arising from casual acceptance of the information without an adequate understanding of the way in which it had been generated, or collected. Howison and Crowston (2004) and Rainer and Gale (2005) both warn against accepting observations based on *SF.net* as representative of the FLOSS community at large, in particular noting the high number of one-person and/or inactive projects hosted by the platform and the absence of many of the largest projects, which have their own web-sites. As has been noted previously, however, this study treats *SF.net* and the behaviors of the individual registrants not as representatives or exemplars of the universe of distributed collective production communities, but on their own terms, as reflecting the experience of a quantitatively important (but nonetheless particular) collaborative development environment. There is another "pitfall" to be borne in mind, which results from the platform's "open-ness": projects that have been largely developed externally to the *SF.net* environment, even those whose code is hosted elsewhere, may be posted on *SF.net* in order to give them greater visibility (Comino *et al.*, 2005; Howison and Crowston, 2004). Where such is the case, a project that actually was

⁹ Where automated procedures were not able to correct suspect data entries or identify a project founder, each problematic item was corrected by hand, comparing the available information with other sources of data (other records in the database, the *SF.net* website, and in some cases interviews with *SF.net* developers). Some assumptions about specific features of the operation of the *SF.net* data system were unavoidable, but these were both plausible and few in number. The effects of residual errors in the observations are closely limited by the large number of developers involved in most of the processes that are examined in this analysis. A series of controls on specific sub-samples, and the behavior of the aggregate results confirm that any remaining data errors can have only a very minor impact on the results. A further discussion on these points can be found in Giuri *et al.* (2006), that project being the original context in which the dataset was prepared.

important and very much alive (elsewhere), nonetheless, might appear on *SF.net* to be both small and dormant. This however means that the project's relationship with the platform must be very weak, and role of the project in the whole ecology must be marginal. Our data detect precisely these characteristics. Thus, since we focus on the 'window' *SF.net* offers to look at FLOSS development, this problem is to a certain extent irrelevant in our analysis.

2.2. Methodology: Markov chain analysis

Given the nature of the *SFnetDataset*, we need a statistical framework that is able to capture parsimoniously the main features of the micro-level processes that are of interest, namely, resources mobilization and new projects foundation. The application of Markov chain analysis (see e.g., Amemiya 1985, Hamilton 1994) is quite suitable for this purpose, and the appropriate first step is taken by delineating a manageably compact "state space" that describes an exhaustive set of activity-states into which individuals observed in this environment during any interval of time can be classified. People registering on *SF.net* participate in the activities of this "community" in several basic roles:¹⁰ they may "lurk" without interacting with others or with existing projects in ways that the infrastructure of the platform would record; they may send patches, bug reports or request specific features to one or more of the established project, which are recorded; they may enter one or more projects and be listed as "group members;" they may initiate new projects. A scale of activities is implicit in this, ranging from inactive behaviors to the most entrepreneurial ones, and resources mobilization can be conceived broadly as a transition from the lowest to the highest levels of developers' involvements into the community activity. It is useful, however, to recognize three discrete levels in the "mobilization" of resource inputs: (i) contribution of bug reports, patches, and feature requests while remaining external to the projects, (ii) contribution as (internal) members of projects, and (iii) the "founding" of new projects.

Consequently, it is possible to distinguish and capture all of these activity/roles by assigning every registrant to one and only one among the following set of "activity-states" in each of every 30-days period of her or his experience in *SF.net*:

- 0=non member and non founder¹¹, inactive¹²;
- 1=non member and non founder, active;
- 2=member of 1 project and non founder of any project (both active and inactive);
- 3=member of more than 1 projects and non founder of any project (both active and inactive);
- 4=founder of 1 project and member of 1 project (both active and inactive);
- 5=founder of 1 project and member of more than 1 projects (both active and inactive);
- 6=founder of more than 1 project and member of more than 1 projects (both active and inactive).

With the states $s=1, 2, \dots, 6$ belonging to space s thus defined, observations on the frequency of the movements of individuals from state i in "epoch" T_i to state j in the next temporally adjacent "epoch", T_{i+1} , can be used to calculate average state dependent transition probabilities, and the results may then be organized to form a transition probability matrix. Treating these estimated mean probabilities as constant, the system is a homogeneous Markov chain with transition matrix $\mathbf{P}(t) = \mathbf{P}$. A Markov chain is called *ergodic* if the distribution of the system among its states at step r , $\mathbf{p}(r)$, converges to a limiting (or equilibrium) distribution, \mathbf{p} , that is independent of the initial distribution $\mathbf{p}(0)$.¹³ By analysis of the transition matrix \mathbf{P} it is possible to establish whether a homogeneous Markov chain satisfies the conditions for ergodicity, which is

¹⁰ In this portion of the text we have taken care not to refer to all the individuals that are registered on SF.net as "developers", inasmuch as there are many visitors to the platform who left no discernable trace of having interacted with any among the projects. In subsequent discussion of those in the "active state" (beyond state 0), however, the generic designation "developers" is employed.

¹¹ The only "foundations" we take into account are those who generated a project whose founder is still part of it at the end of the month in which they were launched.

¹² The developer is considered active if she did post at least one item classified as "patch", "bug report" or "feature request" to the tracker system of *SF.net* of at least one project during the analyzed period. Otherwise, she or he is considered inactive.

¹³ If the Markov chain is ergodic, the probability of remaining in each state goes to a constant (which is the inverse of the mean recurrence time) at the limit, and the resulting limiting distribution is found as the solution of $\mathbf{p} = \mathbf{pP}$.

to say that the iterated transition process eventually will cause the system to shake free of any arbitrary initial distribution $p(0)$. When that is the case, it is convenient to calculate the limiting (ergodic) distribution of developers among activity states as a way to expose the persisting dynamic forces of the system, abstracting from the effects of transient influences that may have affected the initial distribution of the individuals. In presenting this approach, it is important to emphasize that the limiting (ergodic) states we will exhibit are not interpreted here as long-run forecasts, because the length of the period covered by the *SF.net* archive that is under examination here is quite short, and far too brief to allow meaningful predictions for this evolving environment.¹⁴

The questions that we seek to answer make it necessary to specify the time dimension of the states and the transition process by reference to fixed durations of developers' *experiences* on the platform, i.e. the spans of time elapsed after each individual's initial entry-event – which must be defined independently of the particular calendar dates on which she or he registered on *SF.net*. Further, for the purpose of studying transitions between any pair of “activity states,” it is desirable to compare the positions of the members of a cohort of developers among the activity states in successive “epochs” of uniform duration. Therefore, for a given transition, we divide the experience of each developer between two “epochs” (*A* and *B*) that are of equal absolute length: T_A , corresponding to the days observed on *SF.net* from x to y , and T_B , the days observed from $y+1$ to z , where $y-x = z - (y+1)$. Moreover, in order to obtain estimates of the transition probabilities that form a first-order, or current state-dependent chain, it is important that the duration of those “epochs” not be so brief that the probabilities of any specific transitions from any of the states will be influenced by the individual's position among the states at some more remote, anterior point in time.

To fix the values of x , y and z that bound the two “epochs” for the distributions of individuals among the “activity-states” are to be obtained, it is necessary to consider the temporal structure of the underlying phenomena to which those “summary” states refer. In Appendix 1 we report the considerations that have led to the selection of “standard” observational epochs having durations of 180 days, on the basis of time series analysis of monthly (30-day) observations of events of project-joining, and, similarly of project-founding, for the sub-set of individuals who registered on *SF.net* during the period from September 1, to October 30, 2000. Following this cohort over the ensuing 840 days provides a large enough number of 30-days periods (28) to extract the possible cycles by spectral analysis of the two series -- after filtering to remove the trends.

Among the procedural details of this preliminary data analysis that should be mentioned here, the first derives from our ultimate concern to examine the behaviors of developers who were active in creating or joining projects: the observational sample of individuals in the time-series studies has been restricted to the sub-sample who founded or joined at least 1 project during their first 30-day period on the *SF.net* platform. Doing so serves to remove many passive or marginally active registrants from the series examined by spectral analysis and thereby render more readily apparent the periodicities of the behaviors of joiners and founders. Moreover, this creates a common starting point (the first project launched) for all the individuals, and sets the beginning of the series at the moment of the first foundation, thus employing the whole series time span to describe successive foundation activities. In the case of the time-series of project-launchings, the data from the first two of those 30-day intervals (from day 1 to day 60) were omitted -- in order to reduce the influence of the “importation” of pre-existing projects in the first months of individuals' experience of the *SF.net* platform (see appendix 1 for a discussion on this point) and thereby better reveal such periodicities exist in the “indigenous” process of project-founding on the platform. As the same consideration does not apply with equal force in the case of the time series of project-joining events, the full time series was used in that case.

The upshot of the preliminary data analysis reported in Appendix 1 is, first, that there are indeed periodicities in both kinds of events, and second, that these can be taken into account by defining 6-month

¹⁴ Moreover, even if it could be reasonably assumed that the elements of the transition matrix estimated from the observations in the *SFnetDataset* would remain time invariant, successive distributions of the platform's population among the activity states would be evolving under the disturbing influence of the entry of new participants. A forecast of the latter, possibly non-stationary process would therefore be necessary in order to produce a forecast of the changing distribution throughout what might be an extended quasi-ergodic phase. No such forecasts are made here.

long observational “epochs.” The latter are sufficiently extended to capture completion of most of the important periodic structure of events that drive transitions among the activity-states, yet not so long as to foreclose the opportunity to observe more than one typical cycle and thereby prevent examination of the temporal stationarity of the estimated transition matrix.¹⁵

3. The Transition Probability Matrix, and Comparison of Initial and Ergodic Distributions

In calculating average probabilities of transitions between activity states defined for each pair of consecutive 6-month long epochs, the data from the 222,583 the registrants during their first 90 days on *SF.net* were not used. The same considerations that led to discarding the first 60 days of observations on these individuals activities for the purposes of the preliminary time series analysis (in Appendix 1) apply here: this is intended to filter out the effects of “imported projects” and the associated project members among the newest cohorts on the platform.¹⁶ Observations on activities that took place during the third month on the platform were excluded in the estimation of transition probabilities for a different reason: excluded observations during an “entry period”, from which registrants’ positions among the 7 activity states may be treated as exogenous variables, can be used to stratify the whole population into sub-groups defined by their early activity characteristics, thereby enriching the analysis of project joining and launching behaviors. Since the first two months suffer of the “imported projects” effect, we needed to rely on the third month of registrants’ experience in *SF.net* to stratify the population based upon initial (entry phase) activity-status.

Thus, in estimating the transition probabilities we considered two intervals: epoch *A*, based on observations that span the 30-day months from the 4th to the 9th (so that $x=90$ and $y=270$): epoch *B*, based on observations pertaining to 30-day intervals 10 to 15 (i.e. $z=450$). By considering all 222,835 individuals who registered on *SF.net* during the (approximately) 14 months from September 1, 2000 to October 26, 2001 it was possible to retrieve from the database observations covering each individual’s behavior throughout a 360-day time span. The distribution of the sample among the seven states at the close of period *A* is presented by Table 1, and for the purposes of this computation we refer to this as the population’s “initial distribution”.

Table 1. The “Initial” (Epoch *A*) distribution of the whole sample

<u>State</u>	Frequency	Proportion
0	192051	0.862
1	2202	0.010
2	21848	0.098
3	3280	0.015
4	1702	0.008
5	1405	0.006
6	347	0.002

Entries for the transition probability matrix are obtained using the MLE estimator described by Amemiya (1985). The resulting matrix *P* describing the transition probabilities between the states attained at the end of in epoch *A* (rows) and epoch *B* (columns) is reported in Table 2 and depicted in Figure 1. The shading of the zones in the figure indicates equal probability bands, and the steepness of the gradient between a given band and the one adjacent is approximately represented by the width of the band in question.

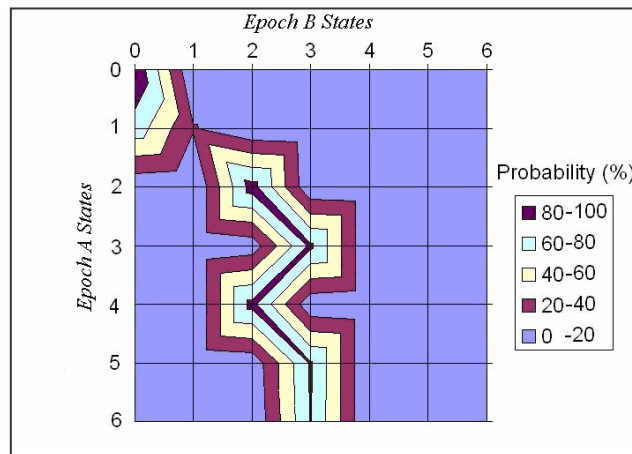
¹⁵ It should be clear that this analysis is undertaken solely for the purpose of minimizing the amount of noise (and possible bias) by imposing arbitrary periods when calculating the frequencies of transitions between pairs of states. There is no intention here of investigating whether it is appropriate to characterize either (or both) of these micro-level behaviors as continuous time Markov renewal processes (see Pyke, 1961; Cox, 1962). For applications of Markov chain theory and renewal models to micro-demographic data, see, for example, Sheps and Menken (1973).

¹⁶ This filter remains imperfect, however, because it is possible that delays in the formal organization and launch of projects formed substantially on the basis of “imported code” may extend beyond the initial three-month period following registration. The need to extensively restructure the architecture of the code of a project that had been previously developed by a closed process, in order to make it more suitable for further development by a decentralized, virtual community, was recognized in the well known case of Mozilla. That is only one of the possible circumstances that, along with decisions about governance and licensing, might contribute to significantly extended delays between the individual registrations by a core of project developers, and the date of the formal launching of their project.

Table 2. The transition matrix P_{AB} for the whole sample of *SF.Net* registrants

		Epoch B States							# developers
		State	0	1	2	3	4	5	
Epoch A States	0	0.9821	0.0072	0.0047	0.0004	0.0046	0.0005	0.0005	192051
	1	0.7134	0.2153	0.0395	0.0023	0.0232	0.0050	0.0014	2202
	2	0.0485	0.0009	0.8892	0.0295	0.0010	0.0275	0.0034	21848
	3	0.0116	0.0000	0.0753	0.8424	0.0003	0.0646	0.0058	3280
	4	0.0100	0.0000	0.8702	0.0470	0.0018	0.0652	0.0059	1702
	5	0.0021	0.0000	0.0498	0.8206	0.0000	0.1060	0.0214	1405
	6	0.0000	0.0000	0.0115	0.8098	0.0000	0.1268	0.0519	347

Figure 1. Representation of the Transition Matrix P_{AB}



It may be seen that there is strong inertia within each of states 0, 2 and 3, indicated by the high probabilities of persistence from one 6-month period to the next, and that state 0 (inactivity) remains a strong temporary attractor even for those in state 1. By contrast, persistence at the higher levels of activity is much rarer: individuals reaching each of the three activity states above 3 (all of which involve founding one or more projects) are very likely not to found any new projects in the following epoch. Instead, the tend to return to lower states than those that they had just attained: 4.7% of those who reach state 4 (on average) will proceed to join another project in addition to the one they founded (state 3), and only 7.29% (on average) will launch at least one other project (states 4, 5 and 6) in the following 6-month epoch. An intuitively appealing inference is that project *launching* involves time-consuming commitments, and therefore is a current activity state that is only very rarely sustained from one 6-month interval to the next.

The local strength of state 0 as an “attractor”, already noted, is another marked asymmetry in the dynamics revealed by Figure 1: not only is the mean probability of exiting state 0 quite small (0.018), but the probability of becoming inactive (state 0) after a period of activity as “non-member” (state 1) is almost forty times larger (0.713). With regard to the mass of those registrants who enter the platform in state 0 and never emerge from that condition of passivity, labeling them as “developers” seems rather problematic. They may well be “developers” who participate in FLOSS projects that are not on *SF.Net*, and so are both interested and expert visitors to “the *SourceForge* development district”. But, they also may be merely curious Internet surfers, or social scientists who had to register in order to gather some data. If the label “*SourceForge* developer” is to have any real content, it would seem necessary to exercise care in its application. Here it is as rule reserved for those who do leave the inactive state. The implication of the transition matrix estimates, however, is that “*SF* developers” are a distinct minority of those who visit this FLOSS collaboration environment: for the average registrant on *SF.net* the probability of a first emergence from state 0 in each successive month is 0.003, and from this it follows that the likelihoods of progressing to project membership and/or project launching (states from 3 to 6) via state 1 (i.e. as an unattached “lurker”) must be quite small.¹⁷

¹⁷ Since the 6-month probability (shown in Table 2) for remaining in initial state 0 is 0.9821, the corresponding monthly

At such rates, within a 6-month interval *only 1 in two thousand registrants* could be expected to make the passage envisaged by the onion model -- from inactivity to peripheral activity and thence to approach the “core” of one or another project.¹⁸

It seems evident that if the onion model has any relevance it pertains to the population of individuals who have some intention of becoming open source developers, whereas the great mass of individuals that register on *SF.net* do so essentially as “spectators”. One may calculate that the probability of becoming a member of a project group, conditional on having not been part of any project in a previous 6-month interval is as small as 0.011. By contrast, once a developer has joined or founded a project, the probability of not returning to any of the activity states below 2 is very much higher, indeed, almost a certainty (0.960). Notice in this connection that formal membership is the criterion used here to place developers in the different projects. Therefore, this means that unless a project member withdrew from the group membership list, or the project itself was formally closed and removed from the official list of “active projects” by the *SF.net* staff, an individual will be counted as being continuing member of the project so long as she or he remained listed as such at the close of at least one month in the period of observation. Consequently, on the basis of data of the present kind, “project membership” should not be interpreted as implying some uniform level of contributed activity either through time, or among projects.

Closer analysis of the data (see Appendix 2) reveals that the MLE estimates of the transition probabilities in Table 2 for the whole sample of 222,835 developers (those registered to *SF.net* from September the 1st, 2000, to October the 26th, 2001) are not describing a Markov process that is strictly time-stationary within the span of observation. To study the question of stationarity we relied on a sub-sample of registrants whose activities could be observed over three consecutive 6-month epochs so that transition probabilities among the states could be computed from Epoch *a* → Epoch *b*, and from Epoch *b* → Epoch *c*. In order to do this and omit the first 90 days of the individuals’ experience on the platform from the analysis, the sample had to be reduced to the 104,698 individuals registering from September the 1st, 2000 to 28th April, 2001. Although comparisons of the MLE estimates of transition matrices P_{ab} and P_{bc} (displayed in Tables A2-1 and A2-2 in Appendix 2) shows that in many respects they are quite similar, there are small but statistically significant differences.¹⁹ In the later period there are higher probabilities that individuals who launch new projects (states 5 and 6) will persist in one or another of that pair of activity states, rather than returning to project membership status (state 3). So there would seem to be some evolution among the subpopulation of “founders” in the direction of more intense project launching.

Other findings may be noted which convey a related message about the emergence of project founders from the ranks of the developers on this platform. When stratifying developers in cohorts according to their activity status during the 90-day “entry period”, it is found that Kolmogorov-Smirnov and Mann-Whitney tests show all the cohorts to have similar initial distributions. Nevertheless, the transition probabilities for different strata are significantly different.²⁰ A plausible explanation for this result is that the growth of the platform brought with it, or induced a change in the composition of the registered users. In comparison with the early wave of registrants, those entering at somewhat dates in *SF.net*’s history appear considerably more likely to “lurk” during their initial months on the platform, but thereafter rapidly become polarized between the extreme states on the continuum ranging from “inactivity” and “high activity.” This invites the conjecture that the distributions among the activity states of the two distinctive patterns of behavior (corresponding to distinct sub-populations of developers) change in ways that are out of phase with

probability of persistence is calculated as the sixth-root, and the mean probability of emerging is found to be 0.003, as reported in the text. This calculation assumes that the 6-month transition rates are generated by constant monthly probabilities.

¹⁸ From table 2 the probability of progressing directly to any of the states 2-6 from state 0 via state 1 is found for 6-month transitions as $(0.007) [1-(0.713+0.216)] = 0.0005$, where 0.007 is the probability of going from state 0 to state 1, 0.713 the probability of remaining in state 1, and 0.216 the probability of returning from 1 to state 0.

¹⁹ The test establishing this (described in Appendix 2) applied the procedure developed by Bickenbach and Bode (2001, p.12), who present a modified version of the test by Anderson and Goodman’s (1957: p. 97).

²⁰ The test proposed by Bickenbach and Bode (2001: p. 8), is an adaptation of the Anderson and Goodman’s (1957: p. 99) that can be used to check for homogeneity with respect to the entry period and other exogenous traits.

each other during the quasi-ergodic stage of the Markov process. Some developers enter *SF.net* and quickly become active, sometimes founding projects which are likely to have been already undertaken outside the platform. After this initial phase of activity, however, they move back to the less active states, just as the developers who initially were passively “lurking” emerge as more active participants within the community and become the sources of the platform’s innovative performance. This latter group’s deferred passage into “activity” and eventually into project founding will be analyzed more closely below, in sections 5 and 6.

4. Project Joining and Project Founding: the Equilibrium of the Iterated Markov Chain

To discern the persistent micro-dynamic processes that are at work within the virtual “district” constituted by *SF.net* requires that it be based on an estimated model of a Markov chain that is stationary, or at least not undergoing very perceptible changes over time. In view of the absence of stationarity in the process described by the transition matrix (Table 2) for whole population of registrants in the *SFnetDataset*, our analysis of the properties revealed by the iterated Markov chain is based on re-estimating the transition probability matrix for the same consecutive 6-month epochs *A* and *B* but using only the data pertaining to members of the last among the cohorts of registrants. The sample population that could be used for this purpose was thereby reduced to the 79,983 individuals whose entry dates fell within the 120 day span extending from June 28th to October 26th, 2001. Using that data, the transition probability matrix obtained by the MLE method is presented in Table 3, where it is denoted as $(P_{AB})^{last}$, thereby distinguishing it from other matrices with which it may be compared.²¹

Finding the limiting distribution implied by these transition probabilities requires that one first establish that $(P_{AB})^{last}$ determines a regular Markov chain, or has an absorbing state. While one may see immediately from Table 3 that there are some zero elements in that matrix (as it was the case also in P_{AB} shown in Table 2), it turns out that all the entries in the third power of the matrix are positive, so that $(P_{AB})^{last}$ is indeed a regular transition matrix and its associated Markov chain is regular. Since regular Markov chains are ergodic, one may readily solve for the limiting distribution (see Amemiya 1985, p. 421), and compare the result obtained with the “initial” distribution of the same cohort -- which appears in the right-most column of Table 3. To better bring out the respects in which the proportions in the 7 activity states of the limiting distribution differ from those initially observed (i.e., in epoch *A*), the two distributions are reported in Table 4 and graphed on a logarithmic scale in Figure 2.

Table 3. The transition matrix $(P_{AB})^{last}$ for registrants from June 28th to October 26th, 2001.

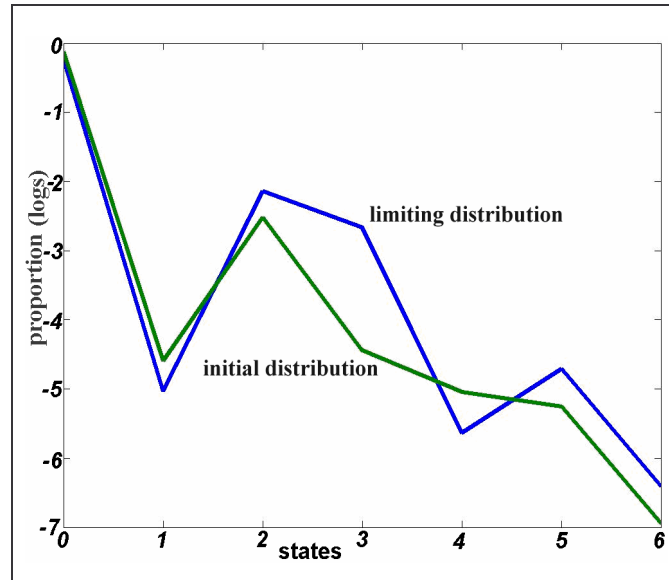
		<i>Epoch B States</i>							
<i>Epoch A States</i>		0	1	2	3	4	5	6	# developers
	0	0.9845	0.0063	0.0037	0.0004	0.0041	0.0004	0.0006	70766
	1	0.7241	0.2143	0.0369	0.0025	0.0172	0.0025	0.0025	812
	2	0.0561	0.0009	0.8882	0.0249	0.0009	0.0266	0.0023	6458
	3	0.0139	0.0000	0.0897	0.8301	0.0011	0.0577	0.0075	936
	4	0.0116	0.0000	0.8702	0.0407	0.0039	0.0678	0.0058	516
	5	0.0000	0.0000	0.0670	0.7847	0.0000	0.1196	0.0287	418
	6	0.0000	0.0000	0.0000	0.7922	0.0000	0.1299	0.0779	77

²¹ The matrix in Table 3 may be compared to the transition matrix estimated from the whole sample (i.e. P_{AB} in Table 2), as well as to the matrix estimated from a still smaller sample of observations in a later entry cohort – namely that for the last two months in the available sequence of “months” (from August 27th to October 26th, 2001). In all cases the Markov transition matrices and the associated Markov chains converge to a unique distribution. While the implied relative speeds are much the same during the initial 40 epochs, the absolute time required to closely approximate the limiting distribution is roughly 20% shorter in the case of the two subsamples capturing the last entry cohorts than it is in the case of the (non-stationary) process estimated for the whole sample population. Eventually, notice that the relationship between the initial and the limiting distributions are very similar for all the three samples.

Table 4. Initial and limiting distributions: developers stratified by their state(s) in month 3 on *SF.net*.

<u>Distribution</u>	<i>State 0</i>	<i>State 1</i>	<i>State 2</i>	<i>State 3</i>	<i>State 4</i>	<i>State 5</i>	<i>State 6</i>
<i>Initial</i>	0.885	0.010	0.081	0.012	0.007	0.005	0.001
<i>Limiting</i>	0.792	0.007	0.118	0.070	0.004	0.009	0.002
<i>L - I</i>	-0.093	-0.004	0.037	0.058	-0.003	0.004	0.001

Figure 2. Initial and limiting distributions for registrants from June 28th to October 26th, 2001.



The comparison between the initial and the limiting distributions reveals the tendency for the overall proportion of lurkers to contract, and for an increasing fraction of developers to occupy states 2 and 3: those remaining in passive states become less prevalent, and the joining of existing projects becomes a more widely diffused behavior. An estimate of the equilibrium “participation rate” for registrants on *SF.net* can be obtained from the limiting distribution in Table 4. In the limit, those who do not belong to any project in the long run represent about 80% of the population, while about 18% of the individuals become part of one or more projects. Only the remaining 1-2% wind up in one state corresponding to new projects foundation. Considering the transition probabilities reported in Table 2 and 3, these results are striking. The “attraction power” of inactive states should be overwhelming, so that in the limit we expect them to absorb all the individuals. On the contrary, the number of developers who continue to create one or even multiple new projects increases over time. This observation, combined with the necessity to evaluate small signals of activity in the light of the dissipative property of the system (see section 1.3) shows that the FLOSS model of innovation is actually able to generate a substantial number of “project founders.”

5. Exploring the Heterogeneity of the Sub-Population of “Project Founders”

Having examined the implications for the limiting rates of project participation and the consequent mobilization of development resources implied by the estimated transition probabilities in Figure 2, we now turn to consider the other interesting set of results, namely those pertaining to the micro-dynamics of project founding. The positive gap that appears between the limiting and the initial proportions in states 5 and 6 means that the underlying dynamics eventually advances a greater proportion of the entering cohort into recurrent project founding activities. In other words, absent other changes, there is a underlying tendency for greater entrepreneurial vigor to emerge, especially among those who become minimally active on the platform.

5.1. Operationally defining the virtual district’s “FLOSS entrepreneurs”

The first step that we can take towards understanding the emergence of *SF.net*’s “FLOSS entrepreneurs” is to examine some objective characteristics of project founders. Fortunately, it is possible to draw upon other studies of the *SFnetDataset* (Giuri *et al.*, 2006, 2005, Rullani, 2006) for data of that kind, as

well as some indicators of characteristics of the particular project(s) on which they were working. The origins of this data are the information collected by the platform from the projects that are listed there.

Project administrators on *SF.net* can post requests for specific development jobs that needed to be done in the context of their respective projects. Counting the frequency of such requests provides a measure of the intensity of those individual's involvement in the project. Alternatively, developers' communicative propensities can be measured by counting the project forum messages that they post. Both measures provide not only static behavioral attributes, but also indicate how developers' involvement (and propensities) change over time. Further, participants in *SF.net* are asked also to fill in and keep updated a form describing their skills in terms of expertise and experience.²² These characteristics have been summarized in a "skills index" composed by the summation of the levels of each developer's skills weighted by their experience. One aim of that undertaking was to capture the effect of skills' level and diversification on the probability of each developer being prepared to perform a managerial vs. more specialized code development tasks in the projects to which she or he joins (Giuri *et al.*, 2005); another purpose was to assess the effect of the skill levels and diversification of capabilities among project members on the survival and performance of the different projects composing the *SF.net* ecology (Giuri *et al.*, 2006). The available data about project members' skills are not unproblematic: not only are they self-declared, but in the present instance it is possible to observe the reported information only at the end of the time span of the dataset that is under analysis (January 2003). Consequently, it must be acknowledged that the mean values of the individual skill indexes that are computed for different sub-groups of developers reflect their respective experiences as *SF.net* project members and project founders and cannot be treated here as though they were exogenous, "fixed effects" describing attributes of the "representative developer"; hence that are useful simply in identifying descriptive patterns, rather than casual influences upon the behaviors of members of those sub-groups²³.

The same must be said in regard to another set of measures that capture characteristics of the project environment(s) to which individual developers are exposed (or choose to be part of). The level of the social communications activity "surrounding" the developer can be quantified as a count of the messages that her or his collaborators post to the forums of the projects to which she or he contributes as a member. The number of CVS commits the individual developer's project groups have produced, the numbers of each developer's co-members in those projects, and the number of projects they have joined, similarly can be used to build a typical "project profile" for each individual. Unlike the skill indexes, these measures of the activity of developers' respective projects (including communications activity) have been calculated for the first period of developers' post-registration experiences in *SF.net*, as well as over the 15 periods referred to in constructing the transition matrix (as described in sections 2 and 3, above). Considering these two different time spans provides a view of the evolution of the founders' characteristics. To properly assess the distinctive attributes of founders, it is useful to define a reference group. In the following, therefore, the sub-population consisting of developers who founded at least one project during periods from 1 through 15 are compared with the larger population of individuals who were members of at least one project within the same time span. This yields the following two sub-samples:

sample m = members of projects during "months" 1 to 15: N=31,460;

sub-sample f = founders of projects during "months" 1 to 15 : N=15,825;

For each group the number of different actions is computed and averaged ($avg(f)$ and $avg(m)$) over the aggregate number of developers and, when necessary, over the number of periods, leading to the results shown in Table 5. The entries in the table's third and sixth columns show the ratio between the averages of these metrics for the groups of founders f and "non-founders" (nf), i.e. $avg(f)/avg(nf)$. The latter relative

²² Developers are asked to classify their skills along five levels of expertise (1="Want to learn"; 2="Competent"; 3="Wizard"; 4="Wrote the book"; 5="Wrote it") and five time periods describing their experience (1="less than 6 months"; 2="from 6 month to 2 years"; 3="from 2 years to 5 years"; 4="from 5 years to 10 years"; 5="more than 10 years").

²³ Both problems, i.e. self-declaration and endogeneity of skills, are well described and tackled in the quoted studies. Notice that there specific techniques are applied precisely to overcome the endogeneity problem.

measure is readily derived from $avg(f)$ and $avg(m)$ by noting that the set of founders is a subsample of the entire set of members and that the sample f is half as large as sample m .²⁴

Table 5. Comparison of average individual and project characteristics for “founders” and “members”

Variable description (average number of actions per developer)	Actions in the first period			Actions in periods 1 – 15		
	$avg(f)$	$avg(m)$	$\frac{avg(f)}{avg(nf)}$	$avg(f)$	$avg(m)$	$\frac{avg(f)}{avg(nf)}$
- skill index (at January 2003)	45.506	43.848	1.078	45.506	43.848	1.078
- forum messages posted by the developer	0.325	0.363	0.810	0.175	0.161	1.190
- jobs required by the developer	0.076	0.039	38.00	0.018	0.009	∞
- average number of members of the projects the developer belongs to	1.251	3.153	0.247	2.018	4.589	0.282
- forum messages posted by other developers to the projects the developer belongs to.	2.838	4.816	0.418	1.380	2.627	0.356
- average number of CVS commits of the projects the developer belongs to	5.741	22.215	0.148	8.893	26.357	0.203
- number of projects participated by the average user populating the average projects the developer belongs to	0.777	0.885	0.782	1.192	1.239	0.927

Inasmuch as launching and continuing to work on a small project seems to be typical behavior for a very substantial portion of the individuals involved in FLOSS development, including many that have only recently entered and become active, it seems unlikely that the individual characteristics of these “founders” would differ very markedly from those of developers who join existing projects rather than launching new ones. This supposition is confirmed by Table 5: not only are the members and founders sub-groups similar in terms of their average skill index and number of messages sent to projects’ forums, but the corresponding ratios of the founders to the non-founder group members are fairly close to unity. It is striking, however, that the average level of communication activity among the founders evolves more rapidly than that of non-founder project members, starting about 20 percentage points below the latter, and ending about 20 percentage point higher – almost a 50 percent gain in relative level. This may reflect a tendency for the general activity levels of project members other than the founder to be more intense in their early phase in the project and to decline their after, whereas the opposite dynamic characterizes the communications of the founders of projects that survive and go on gaining members. These observations, however, hint only rather broadly at dynamic phenomena that can and should be studied more directly both by analysis of project forum and email archives.²⁵

In other respects, these data show expected differences arising from the tendency of founders to belong to projects that start smaller and remain smaller than those to which the mass of non-founder members belong, and form the “role” differentiation implied by taking the responsibilities of being the founder. In fact, founders post “required jobs” whereas non-founder members do so rarely, i.e. most of the time they do not lead the projects they are in. Moreover, the smaller projects of the founders have relatively fewer forum messages, and this difference becomes more pronounced over time, even though there is some tendency for the relative average membership size of founders’ projects to increase *vis-à-vis* the projects of non-founders. The average number of CVS commits per project member start out, in period 1, at a small fraction of that found on the average projects of non-founders, but although this measure of development-activity does show a substantial relative rise over time, it remains comparatively low. A similar conclusion can be drawn for forum messages posted by other developers, even if in this case the dynamics moves in the opposite direction.

²⁴ Being a founder of a project means being part of that project. Thus, sample f is a subsample of sample m . Moreover, population f is composed by approximately half of the individuals belonging to m . So we compute: $[avg(f)]/[avg(nf)] = [avg(f)]/\{2[avg(m)]-[avg(f)]\}$.

²⁵ See Crowston and Howison (2005), for a network analysis approach that could be extended to explore the dynamics of intra-project communications, and identify the activities of central participants.

But even if smaller, less communicative and generating fewer commits per member, the projects to which founders belong do not appear to be so different from others when considering the position they have in the network of interactions among *SF.net* collaborators. Taking the number of projects in which the average member of developer i 's average project participates as a rough proxy for the "centrality" of i 's projects in the network, Table 5 shows the relative difference between founders and non-founders' sub-populations to be less marked than one might have expected, and tending to shrink with the passage of time. Being a founder on *SF.net* therefore does not seem to be associated with any very markedly distinguishing patterns of behavior, other than those that can be attributed to having taking leading roles in comparatively small start-up projects.²⁶

5.2. Effects of entrepreneurs' previous activity on project launching

In the construction of the transition matrix, month 3 was excluded in order to use it as an indicator of each developer's initial propensity to leave the passive lurking state and participate in the activities of the projects on the platform. To capture the level and nature of developers' participation in these early phases, we stratified the sample on the basis of the state attained by each developer during the period ranging from day 60 to day 90 of her or his experience in *SF.net*. States 4, 5 and 6, having as a common condition the founding of at least one new project, are grouped together in order to provide sufficient sample density to permit estimation of transition matrices for each of the identified strata. The outcomes of this analysis, in terms of the resulting estimated Markov chains and their associated ergodic state distributions, are reported in Table 6.²⁷

Consider first the initial distributions of the different sub-samples: Table 6 shows clearly that those who were in state 0, 2 and 3 during the "month" between day 60 and 90 tend to remain in that state during the following 180 days, whereas those who initially belonged to state 1 have a higher probability of moving back to state 0 than of remaining peripherally active. In other words, only about a third of those developers who initially participate in the production of code as external contributors (state 1) keep on contributing, rather than returning to the "passive" lurking state 0 in the following 6-month epoch. Eventually, founding a project in the month between day 60 to day 90 on the platform does not indicate a higher propensity (probability) to launch new projects the ensuing 180 days, even though state 5 eventually "attracts" about 10% of the developers. On the contrary, early founders have a higher probability of subsequently joining projects, as it is seen that they tend to move to states 2 and 3.

²⁶ As said, in the present analysis only those individuals who create a project and are still part of it at the end of the month are considered founders. A part from this minimum requirement, inasmuch as the foregoing comparisons aggregate all founders and make no attempt to distinguish the successful from the unsuccessful projects that they have launched, there is no warrant here for concluding that there are no individual characteristics or qualities of founders that matter, or that would differentiate the successful leaders of new projects from the rest of the developers' population.

²⁷ A potentially worrisome technical complication can be noticed in the results is displayed by Table 5: the emergence of absorbing states. Inaccuracies in the estimated transition matrix –which are likely to arise when, as is the case here, the fine grain of the state space creates specific rows of the matrix with very small number of observations– readily can generate the appearance of absorbing states when the limiting distribution is computed. To assess the robustness of the results presented, a number of simulation experiments were run with arbitrarily modified matrices that mitigated the effects of the uneven distribution of observations in those rows. Even when apparent absorbing states are replaced by weak reflecting states, the qualitative results of the analysis are very similar in many respects, and the interpretation offered in the text still holds.

Table 6. Initial and limiting distributions: developer stratified by their state(s) in month 3 on SF.net

<u>State in month 3</u>	Distrib.	State 0	State 1	State 2	State 3	State 4	State 5	State 6	# devel.	Periods to L
0	<i>Initial</i>	0.971	0.010	0.008	0.001	0.008	0.001	0.001	196998	134
	<i>Limiting</i>	0.786	0.007	0.092	0.096	0.004	0.014	0.001		
	<i>L – I</i>	-0.185	-0.003	0.084	0.095	-0.004	0.013	0.000		
1	<i>Initial</i>	0.576	0.317	0.058	0.009	0.036	0.004	0.000	773	88
	<i>Limiting</i>	0.236	0.034	0.330	0.378	0.006	0.016	0.000		
	<i>L – I</i>	-0.340	-0.283	0.272	0.368	-0.030	0.013	0.000		
2	<i>Initial</i>	0.010	0.000	0.903	0.041	0.000	0.040	0.005	21973	109
	<i>Limiting</i>	0.530	0.005	0.294	0.142	0.010	0.018	0.002		
	<i>L – I</i>	0.519	0.004	-0.609	0.101	0.009	-0.022	-0.003		
3	<i>Initial</i>	0.000	0.000	0.020	0.855	0.000	0.097	0.027	2246	105
	<i>Limiting</i>	1.000	0.000	0.000	0.000	0.000	0.000	0.000	<i>(0 is absorbing)</i>	
	<i>L – I</i>	1.000	0.000	-0.020	-0.855	0.000	-0.097	-0.027		
4,5,6	<i>Initial</i>	0.007	0.000	0.503	0.344	0.000	0.115	0.031	845	2040
	<i>Limiting</i>	1.000	0.000	0.000	0.000	0.000	0.000	0.000	<i>(0 is absorbing)</i>	
	<i>L – I</i>	0.993	0.000	-0.503	-0.344	0.000	-0.115	-0.031		

The implication seems to be that developers’ participation inevitably tends to grow less and less intense as their time on the platform increases. Whatever initial levels of participation they attain, developers are subsequently attracted to states representing lower activity levels. When we assess this effect dynamically, however, the message from the other entries in Table 6 is that the presence in the “lower” states of the distribution of those who belonged to states 0 and 1 in their third month on the platform tends to decrease; whereas the “higher” states of activity tend to become more “populated”.

The picture then turns out to be considerably more complex than examination of the initial distributions would suggest. Not all developers tend to diminish their participation. On the contrary, there is a sort of “role exchange” between different developers, whereby initially active developers tend to participate less over time, and those who are initially inactive “climb” the scale of the states to become members of existing projects or to found new projects. This pattern clearly recalls similar results obtained from the analysis in the previous sections.

5.3. The persistence of entrepreneurial spirits

The previous picture may be enriched by assessing how “persistent” or “durable” is the “entrepreneurial spirit” among those who emerge in the founder’s role on *SF.net*. Consider only those developers registered from September 1, 2000, to October 26, 2001, who founded at least one project in the first 15 periods of their experience on the platform: when a developer in this sub-group is involved at least in a single project, the probability that he or she has founded that project is increasing slightly during the first three months, on average, but decreases thereafter. This fluctuation, however, has a range of only a few percentage points, so that probability can be reasonably viewed as stable in the neighborhood of 0.90. An implication following from this line of considerations is that entrepreneurs stick to the enterprises they have launched, even if for unsuccessful projects this simply means remaining listed as a member of a moribund project while re-directing their energies elsewhere.

By stratifying the sample according to the developers’ founding activity it is possible to further clarify the differential expressions of this “entrepreneurial spirit.” In particular, we may proceed by dividing the whole population (sample **A**, $n=222,835$) into three sub-samples according to the period within which the developers launched their first projects: *subsample B = founders in period 1 – 15, $n=15,825$; subsample C1, founders in period 4 – 15, $n=5,514$; subsample C2 = founders in period 1 – 3, $n=11,875$* . Notice that these groups are not mutually exclusive: developers founding projects in the first 90 days of their experience in *SF.net* and in the subsequent 6 months (about 1500 individuals) will be considered as part of **C1** as well as

of **C2**. Comparing these samples, it is possible to derive some insight on the “role-exchange” feature described above.

We begin by looking at the initial distributions of sub-sample **C2** in Table 7. The group of developers founding at least 1 project in the first 90 days of their experience in *SF.net* is distributed in the following 6 months in a rather different way than the developers belonging to sample **C1**. Part of this difference is due to the definition of the subsamples: **C1** is composed of developers founding projects from period 4 to period 15, and who therefore have greater opportunities to reach states 4, 5 and 6 in greater proportions. But, the Table also reveals --in the high proportion of those developers moving to state 2-- that during periods from 4 to 9 a great proportion of **C2**-type developers do tend to remain part of the project they have founded (or, if they leave the one they have founded, join another project). In other words, their entrepreneurial drive seems largely confined to first projects they launched after arriving on the platform (and which may well have had its origins in an externally formed software package). On the contrary, a high proportion of **C1**-type developers found projects in periods 4-9. In particular, state 4, corresponding roughly to being part only of the project the developer has founded, shows that a consistent portion of **C1**-type developers launch their projects in periods 4-9 without having founded any project during the previous three “months” (or having left the projects they had founded at that time). This marks the difference between the two groups of developers.

Consider now how these initial distributions evolve over time. Developers belonging to sample **C2** are much slower in reaching the limiting distribution than the **C1**-type developers.²⁸ The interesting thing is that when **C1** and **C2**-types are pooled together in sample **B**, the number of iterations needed to reach the limiting distribution is very close to, and even a bit smaller than, the corresponding number for the **C1** subgroup. Even if those developers who founded a project in periods 4-15 comprise half of those who founded a project in periods 1-3, they seem to really be those driving the process, and that would seem to qualify **C1**-type developers for recognition as the cadre responsible for the continuity of the *SF.net* “entrepreneurial spirit”.

Table 7. Initial and limiting distributions according to the period of project launching

Sample	Distrib.	State 0	State 1	State 2	State 3	State 4	State 5	State 6	# devel.	Per. to L
A	Initial	0.862	0.010	0.098	0.015	0.008	0.006	0.002	222835	177
	Limiting	0.727	0.007	0.141	0.107	0.004	0.013	0.002		
	L - I	-0.135	-0.003	0.043	0.092	-0.004	0.007	0.000		
B	Initial	0.074	0.004	0.584	0.119	0.108	0.089	0.022	15825	43
	Limiting	0.004	0.000	0.310	0.573	0.004	0.097	0.011		
	L - I	-0.070	-0.004	-0.274	0.454	-0.104	0.008	-0.011		
C1	Initial	0.193	0.012	0.126	0.042	0.309	0.255	0.063	5514	47
	Limiting	0.001	0.000	0.028	0.436	0.004	0.480	0.052		
	L - I	-0.192	-0.012	-0.098	0.394	-0.305	0.225	-0.011		
C2	Initial	0.009	0.000	0.756	0.153	0.000	0.068	0.013	11875	224
	Limiting	0.140	0.004	0.326	0.460	0.003	0.059	0.008		
	L - I	0.131	0.004	-0.430	0.307	0.003	-0.009	-0.005		

²⁸ It has been found that the transition matrixes pertaining to sub-samples **B** and **C1** both have a single transient state, namely state 1. This jeopardizes the ergodic properties of the associated Markov processes. Nevertheless, the 43rd and the 47th power of those matrixes, respectively, are composed of identical rows (all equal to the limiting distributions shown in table 6) and they remain constant for all successive powers. That assures the existence of a limiting distribution independent of the initial one. This may be seen by considering that for an initial distribution \underline{i} , where $i_1 + \dots + i_n = 1$ (i.e. we express the distribution in proportions), and a transition matrix whose t^{th} power \mathbf{P}^t has all identical rows, the distribution of the process at time t is $\underline{i} \cdot \mathbf{P}^t$. Thus, $l_j = i_1 \cdot p_{1j}^t + i_2 \cdot p_{2j}^t + \dots + i_n \cdot p_{nj}^t$ and since $p_{1j}^t = p_{2j}^t = \dots = p_{nj}^t = p_j^t$, then $l_j = p_j^t (i_1 + i_2 + \dots + i_n)$. Given the definition of \underline{i} , this implies $l_j = p_j^t$ for every j^{th} column of \mathbf{P}^t and entry of \underline{i} . In other words, the j^{th} entry of the limiting distribution is obtained summing the same proportion (indicated by the j^{th} column of the matrix) for all elements of the initial distribution, so that in the limit the proportion of individuals in each state is totally independent of the initial distribution and depends only on the transition probabilities.

Further results in the same vein can be extracted from Table 7. Among C1-type developers, the limiting proportions states 0 and 1 shrink in relation to the initial distribution to a point that the disappearance of “lurkers” is almost complete. C2-type developers, however, tend to be driven in the opposite direction, with the proportion of lurkers starting at a level close to zero and becoming larger in the limiting distribution. Moreover, the greater tendency of this latter group of developers to join existing projects rather than creating new ones is confirmed by the enlarged proportion that tend to wind up in state 3 -- at the expense of state 2. In addition, it may be observed that C1-type developers confirm their “vocation” in that the proportion settling in state 5, at the expense of state 4, reflects their tendency to not only belong to multiple projects but go on launching new ones.

6. “SourceForge Careers”: A Retrospective View of Project Joining and Project Founding

The analysis to this point has sought to describe current activity states and the distributions of the *SF.net* population among them. But career histories and their cumulative achievements also are of interest: individuals’ careers are relevant to their acquisition of experience and reputations, and sometimes to their futures, even in circumstances where “history does not matter” in the strong sense of dynamic processes being “path dependent.”²⁹ In what follows, the focus is explicitly historical or, more properly speaking, retrospective, in that the analysis focuses on the *cumulative* experiences of project joining, and project launching among the developers on *SF.net* during these early years of the platform’s collective history. Although it would be possible to use the current activity transition matrices to generate a stochastic simulation of the distributions of individual careers that are implied by the constant activity-state dependent transition probabilities obtain from the analysis already described, we here opt to construct a simpler descriptive apparatus. This approach defines a new set of “states” in terms of “cumulative career achievement,” and estimates the corresponding transition probabilities for the resulting triangular matrices.³⁰

6.1. Career dynamics: project-founding

Consider first the developers’ distribution with respect to the *cumulated* number of founded projects in the whole time span of the transition matrix (from period 4th to 15th). If the number of founded projects is taken as the measure discriminating among the states we are going to build, aggregating those developers who found more than two projects is a necessary condition to assure that the number of developers “falling” into the states relative to the highest number of founded projects is large enough to yield accurate enough probability estimates. Given this practical constraint, two definitions for the states are possible:

State space 1:

0=founder of 0 projects in periods 4-15;

1=founder of 1 project (p. 4-15);

2=founder of 2 projects (p. 4-15);

3=founder of more than 2 projects (p. 4-15).

State space 2:

0=founder of 0 projects (p. 4-15);

1=founder of at least 1 project (p. 4-15).

State space 1 takes into account the foundation activity at a finer grain, while states space 2 is useful in isolating the passage between being a non-founder and a founder. Accordingly, two different transition matrices have been constructed to describe developers’ movements in these career state spaces. In each matrix the initial state is assigned according to the number of projects developer *i* finds from period 4 to period 9, while her or his state in period is determined by the cumulated number of projects she or he has launched since the beginning of the process, i.e. from period 4 to period 15. Notice that these two periods are precisely epochs *A* and *B* used in section 3. Because this strategy inevitably produces triangular matrixes, it is likely to result in Markov chains that have an absorbing state at whatever is designated to be the highest recognized number of founded projects. Thus, limiting distributions are not relevant in this case. Instead, the

²⁹ See David (2001, 2005) on the formal conceptualization of path dependence in terms of non-ergodic dynamical systems -- a condition that, as has been shown here, does not obtain in the case of the homogeneous Markov chains describing the behavior of the developer population on *SF.net*.

³⁰ Triangularity of the transition matrix is implied by keeping cumulative track of events, which cannot reverse project joining and foundings: progression through the career achievement states, to the extent that it occurs, must be unidirectionally upward.

focus here is on the properties of the process, including the speed with which the overwhelming proportion of initial cohorts of developers eventually reaches the absorbing state.

A point to be noted at the outset is that however the state is defined, and whatever the initial population cohort considered, the Markov chain for processes of this kind generates results with common features in terms of the shapes of curves tracing the distributions among the states over time. In particular, state 0 and the absorbing state have an inverse monotonic behavior, whereas all the other states initially increase their proportion of developers and then slowly decrease it towards 0. But those commonalities aside, the results of the analysis are quite different in terms of the initial distributions, their subsequent values, the number of iterations at which the highest proportion of developers is reached in each of the states, and the total number of iterations needed to reach the equilibrium where all the developers are in the absorbing state. These properties of the evolving distributions are summarized in Table 8.

Table 8. Evolution of the distributions (number of projects founded): two alternative state spaces.

State space	Sample	Property	States				Iterations to equilibrium.
			0	1	2	3	
1	A (whole population, n=222835)	Init. Distribution	0.984	0.014	0.001	0.000	749
		Value of the max	0.984	0.069	0.044	1.000	
		Max at iteration	0	23-24	32-35	749	
	B (founders in period 1 – 15, n=15825)	Init. Distribution	0.782	0.196	0.017	0.005	86
		Value of the max	0.782	0.458	0.227	1.000	
		Max at iteration	0	6	12	86	
	C1 (founders in period 4 – 15, n=5514)	Init. Distribution	0.374	0.563	0.049	0.014	78
		Value of the max	0.374	0.766	0.275	1.000	
Max at iteration		0	1	7	78		
C2 (founders in period 1 – 3, n=11875)	Init. Distribution	0.918	0.068	0.010	0.004	136	
	Value of the max	0.918	0.204	0.108	1.000		
	Max at iteration	0	9	14	136		
2	A (whole population, n=222835)	Init. Distribution	0.984	0.016			733
		Value of the max	0.984	1.000			
		Max at iteration	0	733			
	B (founders in period 1 – 15, n=15825)	Init. Distribution	0.782	0.218			38
		Value of the max	0.782	1.000			
		Max at iteration	0	38			
	C1 (founders in period 4 – 15, n=5514)	Init. Distribution	0.374	0.626			1
		Value of the max	0.374	1.000			
Max at iteration		0	1				
C2 (founders in period 1 – 3, n=11875)	Init. Distribution	0.918	0.082			124	
	Value of the max	0.918	1.000				
	Max at iteration	0	124				

Comparing the number of iterations the four samples cohorts considered below require to reach the absorbing state, a first conclusion is that those who found a project in the periods between 4 and 15 “move faster” than all the other groups. In the case of state space 1, the number of iterations (78) after which this group (C1) as a whole would have founded 3 or more projects is much smaller than the number needed by the whole population (749); and it is smaller than the number of iterations required by group C2 (136). Moreover, the representative individual in group C1 seems to develop greater “career momentum”: when C1

and **C2** are pooled together in **B**, the iterations required to reach the absorbing state decrease from 136 to 86, thus moving much closer to 78. This finding parallels the results obtained in the previous section, namely that those who founded projects closer to the beginning of their time on the platform turn out to be less likely to found new projects in the future, both in absolute terms, and *vis-à-vis* developers whose initial project launches were deferred. Notice too that the results for **C1** under the definition of state space 2 -- in which only one iteration put the group in the absorbing state (of one or more projects joined) -- is not really very informative, as it would appear to be merely a consequence of way that group **C1** has been defined. That is not the case for the developers in sample **C2**: the projects they have founded in periods 1-3 register them simply as members in periods 4-15.³¹

Thus, the comparison between the properties of the processes in the two state spaces sheds light on the two main phases of the founding activity, i.e. becoming a founder, and going on to founding more than a single project, and it helps in understanding the main features of the passage from the first phase to the second. Given this, what Table 8 shows is the difference in behavior with regard to the two phases of project founding: with state space 2 it can be seen that those who found at least one project in the periods 1-3 (**C2**) are more likely to subsequently be in the status of non-founders (state 0) within the time span of the analysis starting with their 4th period of experience; but when they do progress to founding a project they do so *en mass*. This can be seen from the difference between the 124 iterations needed to reach the equilibrium in state space 2, where every developer has founded at least one project, and the 136 iterations needed to bring every developer to the level of having founded more than 2 projects. That difference is much smaller than the one observed for **C1**-type developers (78 *vis-à-vis* 1).

It is informative to examine these processes still more closely. As far as state space 1 is concerned, the maximum level of state 2 is reached earlier and with higher proportions of developers in the case of subsample **C1**. This is due to the specific definition of the subsample **C1**, which requires developers to have founded at least one project in the periods from 4 to the 15. But that “bias” is not off-set by a faster approach to the equilibrium: both sub-samples **C1** and **C2** reach the maximum of state 2 in one-tenth of the total number of iterations needed to arrive at the absorbing state. As before, **C2**-type developers seems once again slower in climbing the stairs of “serial founding” (i.e. having repeatedly launched new projects) than developers belonging to sample **C1**.

In sum, the foregoing analysis confirms this simple characterization of the difference between the career dynamics of sub-samples **C1** and **C2**: that those who launch projects in periods 1-3 are less likely to start new projects thereafter, and the pace of subsequent project launching among them is much slower than that among those who waited until after period 3 to found a project. Yet, although those who are quicker to start a project spend most of their time after period 4 as project members or lurkers, they attain “serial founder” status during periods 4-15 more rapidly on average than those whose project launching career began within that time span. So, this turns out to be yet another story of the early-starting turtles over-taking the hesitant hares in a cumulative career contest: for the former, the total time span required to become a multiple founder of *SourceForge* projects is shorter not because they quicker to “pile up” new projects, but because their launching activity proceeds more regularly from the outset, albeit it at a slower average pace.

6.2. Career dynamics: project-joining

Consider now the evolution of the *cumulated* number of projects joined. The necessary condition to have enough observations in each state is that the state with the maximum number of project participations aggregates all the developers belonging to 5 or more projects. Thus, two possible states spaces are:

³¹ Recalling the specifics of the two alternatively defined state spaces will serve to further clarify this point. The first 4-state space includes one state in which developers do not found any projects and 3 states that distinguish the cumulative (positive) numbers of projects that were founded. With the latter set-up, the Markov chain process reaches equilibrium when all the developers have founded more than 2 projects in the observed course of their experiences on *SF.net*. Under the definition of state space 2, by contrast, the focus falls upon the passage from being a non-founder to having founded at least one project. In this case the process ends when all the developers have moved from the first situation to the second, i.e. when all of them have become founders.

State space 3:

- 0=member of 0 projects in periods 4-15;
- 1=member of 1 project (p. 4-15);
- 2=member of 2 projects (p. 4-15);
- 3=member of 3 projects (p. 4-15);
- 4=member of 4 projects (p. 4-15);
- 5=member of more than 4 projects (p. 1-15).

State space 4:

- 0= member of 0 projects (p. 4-15);
- 1= member of at least 1 project (p. 4-15).

Applying both definitions to the whole population (sample A), the resulting transition matrixes lead to the Markov chains whose dynamic properties are summarized in Table 9.

Table 9. Evolution of the distributions (number of projects joined): two alternative state spaces.

<u>State space</u>	Property	States						Iterations to equilibrium
		0	1	2	3	4	5	
3	<i>Init. distribution</i>	0.872	0.106	0.016	0.004	0.001	0.001	631
	<i>Value of the max</i>	0.872	0.114	0.051	0.031	0.026	1	
	<i>Max at iteration</i>	0	12	22-23	27-28	30-32	631	
4	<i>Init. distribution</i>	0.872	0.128					603
	<i>Value of the max</i>	0.872	1					
	<i>Max at iteration</i>	0	603					

First of all, one should notice that the process of project-joining is faster (in terms of number of iterations) in bringing the whole population into its absorbing state than was the case for the process of project founding, although this result must be carefully interpreted as the two absorbing states are in a sense incomparable: project founding are a proper subset of project joining, inasmuch as every founder, by definition, is recorded as having joined the launched projects. Thus, only a portion of those who move from state 0 to the other states in Table 9 are also among those who make the same transition in Table 8, even if the population is the same. Given this, the observed difference in the transversal rate of approach to the respective absorbing states is to be expected. What is more interesting here is that a wider difference in the mean speeds of the approach to the absorbing states might have been anticipated, while the number of iterations needed to reach the equilibrium in the case of the launching activity are only one-sixth greater than those needed in the “membership” case. In this respect, the two processes do not seem to structurally differ.

Another result along the same path is obtained by comparing the differences between the two states spaces in the case of “joining” and “launching” projects: the proportionate difference between the iteration counts needed to reach the equilibrium in the first case, $(631-603)/603 = .046$ is not so much greater than the corresponding difference in the second case, $(749-733)/733 = 0.22$. What is highlighted here is the importance of the transition from non-activity to participation at some level of activity: once that occurs, developers are more or less on a par in ascending the respective scales of career achievement, both in terms of project-joining and project founding.

Thus the two processes do not appear to be so different in these aspects of their dynamics. If the joining activity is taken to be the benchmark for a minimum level of FLOSS developers’ participation, the results depicted above underscore the relative importance of the launching activity. In a world where dissipation is a prevalent tendency, the entrepreneurial spirit of the *SF.net* virtual district is far from being inconsequential even though the recurrent project founders are a numerically small core within the mass.

6.3. Some implications for entrepreneurship and sustainability

The majority of *SF.net* registered developers remain in the lowest activity states. Of those who are minimally active, the greater portion somehow launch a project during the early phase of their present on the platform, and thereafter become absorbed with that project, or another project that they have not founded. Neither of these groups contributes notably to sustaining the entrepreneurial spirit of the virtual district.

There is a cadre of developers that behave differently, however: they spend a longer time after registering in lurking, and external contacts with existing projects, after which they emerge with a marked higher propensity to create new projects. This is not a sudden process, and takes a while for the transition to project founder status to occur, but once that transition has been taken, it continually fuels the growth of the community.

From the present analysis it is not possible to establish whether this reflects the effects of “involvement processes” at the micro-level, or whether the rising average propensity to launch projects results from the attrition of previous founders who become absorbed in the work of a growing, successful project; and also of those more readily discouraged by the failure of previous ventures to attract supporting resources. Nevertheless, it should be remarked that the “involvement process” hypothesis that would account for the gradual rise in the average propensity for “entrepreneurial actions” (project launching) can draw upon a related study of the *SF.net* community for some ancillary empirical support. Rullani (2006) reports that the more intensely developers are exposed to the social environment of a FLOSS project community, the greater is the number of new projects they may be expected to launch. This observation lends itself readily to an “involvement” interpretation, is quite consistent with the findings from the Markov chain model presented here.

The foregoing lines of explanation concur in implicating the *SF.net* collaborative development environment as having a contributory effect to the sustainability of open source software production, in one way or the other, and possibly both. On one view, the platform’s key function is that of providing opportunities to acquire experience-based capabilities for starting new projects, and stimulating developers’ participation and “entrepreneurial spirit” immersing them in lively environment where other innovators congregate to found new projects. On the other view, there are ideas for new projects aplenty, and the important role of the platform is the rich environment of information, skills, and experience based know-how about the development of open source software. These assist those with new ideas in deciding which of them they should take to the launch stage, and increase their chances of mobilizing others to contribute to the new project. *SourceForge* contributes to this by providing a search space, and search tools where potential project creators can more readily gather information about innovations in software design, and discover others with skills and experience that would be well matched to help turn their ideas into realities.

7. Conclusions, qualifications and opportunities for further development

This study has presented a series of empirical findings exposing the quantitative dimensions of two fundamental micro-level processes at work in the *SF.net* environment: the mobilization of development resources reflected in “project joining” and the creation of novelty reflected in “project founding.” This pair of dynamic processes at the micro-level are fundamental components of any system of innovation, of which the FLOSS mode of production should be considered to represent a particular type, and the attributes of these processes may be said to characterize the basic conditions affecting the regime’s long-run sustainability as an innovation model. A system in which innovators launch enterprises that fail to mobilize others to work on them, or a system in which existing projects and developers migrate to a locale but no new undertakings are created there, must be renewed by external resource injections. Given the relative size and prominence of the *SF.net* platform in the universe of FLOSS activities, the findings based on its performance in these dimensions are not without interest for their bearing upon the question of the sustainability of this larger phenomenon, and the long run development of the system of community peer-based production of which FLOSS is seen by many contemporary commentators to be a paradigm.³²

SourceForge, regarded as an “industrial district,” has been seen here to be more than simply an agglomeration that is attracting enterprises. Instead, it exhibits the property of augmenting the localization of mutually supporting enterprises by generating new enterprises, the property that characterizes an agglomeration that is also an “innovative cluster.” Both in respect to this *exploration* side of the innovation process, and the *exploitation* side that depends upon mobilizing a pool of developers contributing to the

³² E.g., Weber (2004), and more recently Benkler (2006:chs. 2-4). On the question of the long-term success and viability of open source software as a mode of production, one should also consult the software engineering perspectives presented in the contributions to Feller et al. (2005), and particularly the concerns discussed by Brian Fitzgerald (2005) in Chapter 5 of that volume. See also the brief treatment in David (2006).

projects that have been created, these very basic processes for a sustainable system of software production have been found to be robustly present on the *SourceForge*, even though the supportive environment of this platform appears not to be important enough to hold many among the largest and most well known among the FLOSS project communities.

The resources mobilization process and the developers' entrepreneurial spirit were analyzed by building a series of Markov chains from the observation of 222,835 developers' activity on *SF.net* during 2001 and 2002. This methodology has been chosen not with the aim of producing long-run predictions, but rather as an instrument allowing us to look at the deeper dynamic patterns of motion among the various activity states that developers can occupy, abstracting from the transient influences of the initial conditions that surrounded their respective entrances on the platform. Studying the limiting distributions among the activity states that are generated by the transition probability matrices reveals that in the collaborative development environment represented by the *SF.net* platform, the system tends to raise frequency of project joining and project founding activities, including the founding of multiple projects. Moreover, examination of the quasi-ergodic behavior of the project-founding process indicates that among the small group of developers that exhibit particularly pronounced "entrepreneurial" spirits, the average propensity to create new projects remains relatively low during the first stages of their experience on *SF.net*, but subsequently undergoes an increase which sustains the creation of new projects, and even multiple project launching activity. By contrast, those who dominate the ranks of new-projects launchers soon after arriving on the platform, are particularly likely thereafter to drop down to the lowest level of activity.

While the latter of those two patterns suggests that the collective development environment during these early years of *SF.net*'s history was indeed an attractive locale which drew in many software projects that had been created elsewhere, and that were being "re-started" under open source licenses, the former pattern points clearly to the existence of self-sustaining auto-generative forces. True, the "project founders" whose activities constitute the critical *exploration* side of the FLOSS innovation process, represent only a small portion of the developers who belong to project groups on *SF.net*. But, entrepreneurs are always "few among the many". In this instance the fact of their persistence in project launching activities, taken in conjunction with the "open source way of working" to induce project joining by the tens of thousands of developers who enter this platform, should go some way to allaying doubts about the sustainability of the FLOSS mode of collective innovation – at least on these very basic grounds.

Our analysis approaches both the new project generating and resource mobilization processes from the vantage point of the individual actors, rather than that of the projects. This is a significant limitation, reflected in the fact that this study has not dealt with the characteristics of the projects that attract members, or the pattern of circulation of developers among projects of differing sizes, purposes, or governance arrangements. "Project joining" is the outcome of both individuals' propensities for involvement in FLOSS development work of different kinds, of the relative strengths of motivations and preferences for participation in collective community undertakings and independent efforts, on one side.³³ On the other side, it reflects the relative abilities of different projects – and the collective capacity of the ensemble of projects that embrace the generic model of the FLOSS way of working – to mobilize the potential pool of development resources. The picture we have been able to present here is therefore quite one-sided, inasmuch as it has implicitly ascribed participation and project joining behaviors to the individuals, conditioning these on the anterior states they occupied; and, in so doing, has not considered the possible effects of time varying influences arising from the evolving distribution on the *SF.net* platform of projects having different characteristics, and consequently differential attractiveness – both vis-à-vis one another, and the external world of FLOSS projects.

Similarly, our analysis of "project founding" has been restricted to describing a propensity, and examining its association with the distinctive attributes of the founders that could be identified from other dimensions of their behaviors within this particular environment. Although our interpretation of the finding of strong positive association between social communication propensities and project founding is careful to avoid ascribing causality, and leaves it an open question as to whether or not intense involvement in social

³³ The latter are the subject of an extensive empirical and speculative discussions in economics and related social sciences, concerning the nature of the motivations of participants in FLOSS development activity, creating a literature to which both authors have contributed, but into which we have quite deliberately avoiding entering on this occasion. A significant portion of it can be accessed readily from: <http://opensource.mit.edu/>.

communications is an endogenous, learned characteristic arising from the experience of involvement in FLOSS community production activities, other aspects of experience that may shape the behavior of founders have not been recognized in this discussion, much less analysed. In principle it would be feasible, for example, to expand the state space of our Markov chain to consider the dependence of the probability of founding a second project on the “success status” of the individual’s first project, and so on. This extension of the framework would shed light on the issue of whether individual’s are impelled to start new projects by the failures of their previous efforts, or encouraged to try to repeat previous successes in mobilizing support from others and creating an active project.

Quite clearly the Markovian framework for studying FLOSS production communities can be elaborated in ways that would enrich the analysis and provide greater insights into micro-level dynamics that have been dealt with here only in a partial, essentially descriptive fashion. Moreover, the availability of the *SF.net* archives covering the years since 2003 opens up the possibility of enquiring into the stationarity of these processes, or alternatively considered, the evolution of this particular FLOSS ecology. It is to be hoped that the results of this exploratory study will encourage others to pursue and improve upon the beginning that has been made here.

References

- Amemiya, T. (1985), *Advanced Econometrics*. Basil Blackwell: Oxford.
- Anderson, T. W. and L. A. Goodman (1957), 'Statistical inference about Markov chains,' *Annals of Mathematical Statistics*, 28(1), 89-110.
- Baxter, M. and R.G. King (1999), 'Measuring business cycles: approximate band-pass filter for economic time series,' *Review of Economics and Statistics*, 81(4), 575-593.
- Becattini, G. (2001), *The Caterpillar and the Butterfly. An Exemplary Case of Development in the Italy of the Industrial Districts*. Le Monnier: Florence.
- Benkler, Y. (2006), *The Wealth of Networks: How Social Production Transforms Markets and Freedom*, New Haven: Yale University Press.
- Bickenbach, F. and E. Bode (2001), 'Markov or not Markov - This should be a question,' Working Paper, No. 1086, Kiel Institute of World Economics.
- Comino, S., F.M. Manenti and M.L. Parisi (2005), 'From Planning to Mature: on the Determinants of Open Source Take Off,' Discussion paper, No. 2005/17, Università degli Studi di Trento.
- Cox, D. R. (1962), *Renewal Theory*, New York: Barnes and Noble.
- Crowston K. and J. Howison (2005), 'The social structure of Free and Open Source software development' *First Monday* 10(2). Available from http://firstmonday.org/issues/issue10_2/.
- Dalle, J.-M. and P.A. David (2005), 'The Allocation of Software Development Resources in 'Open Source' Production Mode,' in J. Feller et al. (eds.), *Making Sense of the Bazaar: Perspectives on Open Source and Free Software*. MIT Press: Cambridge MA. Preprint available at: <http://siepr.stanford.edu/papers/pdf/02-27.pdf>.
- Dalle, J.-M., P.A. David, R. A. Ghosh and W. E. Steinmueller (2005), 'Advancing economic research on the free and open source software mode of production,' in M. Wynants and J. Cornelis (eds), *How Open Will the Future Be? Social and Cultural Scenarios based on Open Standards and Open-Source Software*. VUB Press: Brussels. Preprint available at: <http://siepr.stanford.edu/papers/pdf/04-03.html>.
- David, P.A. (2006), 'A Multi-dimensional View of the "Sustainability" of Free & Open Source Software Development. Sustaining Commitment, Innovation and Maintainability with Growth,' Revised version of paper presented at the OSS Watch Conference on Open Source and Sustainability. Saïd Business School, Oxford. April 10-12. Revised version available at: <http://www.oss-watch.ac.uk/events/2006-04-10-12/presentations/pauldavid.pdf>.
- David, P. A. (2005). "Path Dependence and Historical Social Science: An Introductory Lecture," An Invited Lecture to the Symposium on "Twenty years of path dependence and Qwerty-effects," convened at the Russian University-Higher School of Economics, Moscow, 13 May, 2005. SIEPR Policy Paper No. [04-022](#) (May 2005) available at: <http://siepr.stanford.edu/papers/pdf/04-22.pdf>.
- David, P. A. (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. Ioannides. Cheltenham, Glos.: Edward Elgar.
- David, P. A. and J. L. Rosenbloom (1990), "Marshallian Factor Market Externalities and the Dynamics of Industrial Localization," *Journal of Urban Economics*, 28: 349-370.
- David, P. A., D. Foray and J.-M. Dalle (1998), "Marshallian Externalities and the Emergence and Spatial Stability of Technological Enclaves," *Economics of Innovation and New Technologies* (Special Issue on Economics of Localized Technical Change, ed. C. Antonelli), 6 (2&3): 147-182.
- Feller, J., B. Fitzgerald, S. A. Hissam and K. R. Lakhani, eds. (2005), *Perspectives on Open Source Software*. Cambridge, MA: MIT Press.
- Fershtman, C. and N. Gandal (2004), 'The Determinants of Output Per Contributor in Open Source Projects: An Empirical Examination,' Discussion paper, No. 4329, CEPR, London.
- Fitzgerald, B. (2005), 'Has Open Source a Future?', Ch. 5 in J. Feller, et al., eds. (2005), *Perspectives on Open Source Software*. Cambridge, MA: MIT Press.
- Giuri, P., M. Ploner, F. Rullani and S. Torrìsi (2006), 'Skills, Division of Labor and Performance in Collective Inventions. Evidence from the Open Source Software', LEM working paper series, 2004/19, July. Available from <http://www.lem.sssup.it/WPLem/files/2004-19.pdf>.

- Giuri, P., M. Ploner, F. Rullani and S. Torrisi (2005), 'Skills and Division of Labor in an Ecology of Floss Projects: Implications for Performance', paper presented at DRUID Tenth Anniversary Summer Conference, Copenhagen, Denmark, June 27-29. Available from <http://www.druid.dk/ocs/viewpaper.php?id=501&cf=3>.
- Giuri, P., G. Rocchetti and S. Torrisi (2002), 'Open source software: from open science to new marketing models an enquiry into the economics and management of open source software,' LEM working paper series, 2002/23. Available from <http://www.lem.sssup.it/WPLem/files/2002-23.pdf>.
- Hamilton, J. D. (1994), *Time Series Analysis*. Princeton, NJ: Princeton University Press
- Herraiz, I., G. Robles, J.J. Amor, T. Romera and J.M. González-Barahona (2006), 'The processes of joining in global distributed software projects,' *Proceedings of the 28th International Conference on Software Engineering*, Shanghai, China, 20-28 May 2006. Available from http://libresoft.urjc.es/Contact/index_html.
- Howison, J. and K. Crowston (2004), 'The perils and pitfalls of mining SourceForge,' in *Proceedings of Workshop on Mining Software Repositories*. International Conference on Software Engineering: Edinburgh, Scotland. May 25.
- Howison, J., M. S. Conklin and K. Crowston (2005), 'OSSmole: A collaborative repository for FLOSS research data and analysis,' in *Proceedings of the 1st International Conference on Open Source Systems (OSS)*: Genova, Italy. July 11-15.
- Hunt, F. and P. Johnson (2002), 'On the Pareto distribution of SourceForge projects,' in *Proceedings of the Open Source Software Development Workshop*: Newcastle, UK. February 25-26, pp. 122-129.
- Jensen, C. and W. Scacchi (2005), 'Modeling recruitment and role migration processes in OSSD projects,' in *Proceedings of 6th International Workshop on Software Process Simulation and Modeling*: St. Louis, USA, May 14-15.
- Krishnamurthy, S. (2002), 'Cave or Community? An Empirical Examination of 100 Mature Open Source Projects,' *First Monday* 7(6). Available from http://firstmonday.org/issues/issue7_6/.
- Lécuyer, C. (2005), *Making Silicon Valley: Innovation and the Growth of High-Tech, 1930-1970*. Cambridge MA: MIT Press.
- Lerner, J. and J. Tirole (2004), 'The Economics of Technology Sharing: Open Source and Beyond,' NBER Working Paper, No. 10956, December, Cambridge, MA.
- Lerner, J. and J. Tirole (2005), 'The Scope of Open Source Licensing,' *Journal of Law, Economics, and Organization*, 21, 20-56.
- Madey G., V. Freeh and R. Tynan (2004), 'Modelling the free/open source software community: A quantitative investigation,' in S. Koch (ed.), *Free/Open Source Software Development*. Idea Group Publishing: Hershey, PA.
- Maggioni, M. A. (2004), 'The Dynamics of Open Source Software Communities and Industrial Districts: the Role of Market and Non-Market Interactions,' *Revue d'Economie Industrielle*, 107, 127-150.
- Mateos Garcia, J. and W. E. Steinmueller (2003), 'The Open Source Way of Working: A New Paradigm for the Division of Labour in Software Development?,' Working Paper, No 1(January), SPRU, Science and Technology Policy Research, Open Source Movement Research, INK: Brighton. Available from <http://www.sussex.ac.uk/spru/publications/imprint/sewps/sewp92/sewp92.pdf>.
- Maurer S. M. and Scotchmer S. (2006), 'Open Source Software: The New Intellectual Property Paradigm,' NBER Working Paper, No. 12148, March, Cambridge, MA.
- Nelson, C. R. and C. I. Plosser (1982), 'Trends and random walks in macroeconomic time series,' *Journal of Monetary Economics*, 10, 129 - 162.
- Pyke, R. (1961), "Markov renewal processes: Definitions and preliminary properties," *Annals of Mathematical Statistics*, 32, pp. 1231-1242.
- Rainer, A. and S. Gale (2005), 'Evaluating the Quality and the Quantity of Data on open Source Software Projects,' in *Proceedings of the 1st International Conference on Open Source Systems (OSS)*: Genova, Italy. July 11-15.
- Raymond E. (1998a), 'Homesteading the Noosphere,' *First Monday*, vol. 3, number 10, October, at http://www.firstmonday.dk/issues/issue3_10/raymond/#d12
- Raymond E. (1998b), 'The Cathedral and the Bazaar,' *First Monday*, vol. 3 number 3, March, at http://www.firstmonday.dk/issues/issue3_3/raymond/index.html
- Robles, G. and Gonzalez-Barahona J. M. (2006), 'Geographic Location of Developers at SourceForge,' in *MSR 2006 International Workshop on Mining Software Repositories*, ICSE 2006: Shanghai, China, May 22-23.

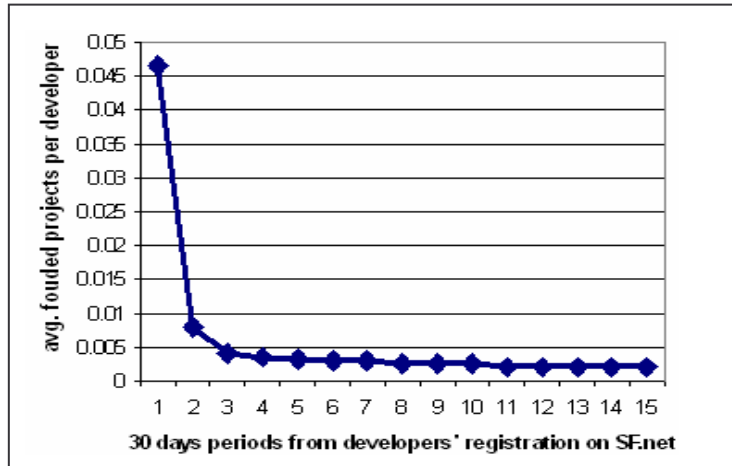
- Rullani, F. (2006), 'Dragging developers towards the core. How the Free/Libre/Open Source Software community enhances developers' contribution', presented at the CCC Thirteenth Annual Colloquium for Doctoral Student Research, May 19 - 21, 2006, Lausanne, Switzerland.
- Saxenian, A. (1994), *Regional Advantage: Culture and Competition in Silicon Valley and Route 128*. Harvard University Press: Cambridge, MA.
- Sheps, M.C. and J. A. Menken (1973), *Mathematical Models of Conception and Birth*, Chicago: University of Chicago Press.
- von Krogh, G., S. Spaeth and K. R. Lakhani (2003), "Community jointing, and specialization in open source software innovation: A case study," *Research Policy*, 32: pp. 1227-1241.
- Weber, S. (2004), *The Success of Open Source*. Cambridge, MA: Harvard Business School Press.

Appendix 1: Setting the Time Constants of the Markov Chain

Launching new projects

Figure A1-1 shows the average number of projects founded in all the successive 30-day intervals within the period of observation by the 222,835 individuals in the sample. The peculiar importance of the period immediately following entry to *SF.net* is apparent.

Figure A1-1. Average Number of founded projects per developer in each ‘month’



A simple explanation of this can be given exploiting again the metaphor of the district described above. Regions are in fact not only areas where the autochthonous resources are mobilized, but also attractors of already established enterprises. Existing firms can decide to relocate in those areas simply because they see the opportunities generated by the agglomeration economies taking place there. Similarly, platforms where FLOSS is developed can be attractors for already existing projects. Developers and team leaders can decide to move the projects they are working on to the platform in order to benefit from the agglomeration economies offered by the platform itself. When this is the case, we observe developers entering *SF.net* and suddenly found a project. The project is not new in itself, but only to the platform. This practice is relatively diffuse. Figure A1-1 suggests that developers are likely to enter *SF.net* bringing with them the projects they were carrying on *independently* of *SF.net*, particularly in the earlier phase of the platform's history. In this case, the foundation activity is not a proper one, and should not be considered in our analysis.

Analyzing the behavior along all the 28 ‘months’ of their experience in *SF.net* of the sub-sample of 1289 developers registered from 1st September to 30th October 2000 who also founded a project in the first period does not change the results.³⁴ The average number of founded project per 30-day ‘month’ is lowered from 1.071 to 0.049 in the first 60 days, and thereafter decreases only slightly, oscillating between 0.026 and 0.006. Thus, in order to detect the cycle, we need to focus on this last series dropping the first and second months. The Dickey-Fuller test on the resulting series shows that it is stationary only at the 5% level even when allowing for a constant. To cut every minimal trend, the series is detrended. Since different de-trending methods are known to be capable of inducing different results, we use the two most common techniques: differencing and polynomial trend removal (see, among others, Baxter, King, 1999; Nelson, Plosser, 1982). Considering the commonalities between the series resulting from first-differencing (**DS1**) and the series resulting from the removal of the linear trend (**POLY1**) enables us to avoid the bias tied to the choice of a specific detrending method. Both first difference and linear trend result in a stationary series, as shown by the Dickey-Fuller test (at the 1% level). Eventually, the spectral analysis of both the detrended series is computed and the relative peaks are ranked in terms of their importance, as shown in Table A1-1.

³⁴ In determine the cycles of projects launching we compute how many projects each user has founded every 30 days regardless of the situation at the end of each ‘month’, i.e. we do not take into account whether the founded projects are still “alive” nor if the founder is still part of them at the end of the 30 days. Notice also that in the last of the observed 30-day periods (the 28th ‘month’) the number of observations drops, due to the fact that the individuals registered in month 2 cannot be observed in their 28th period, but we retained this observation to extend the time series analysis as fully as was possible.

Table A1-1. Cycles of Project Foundings

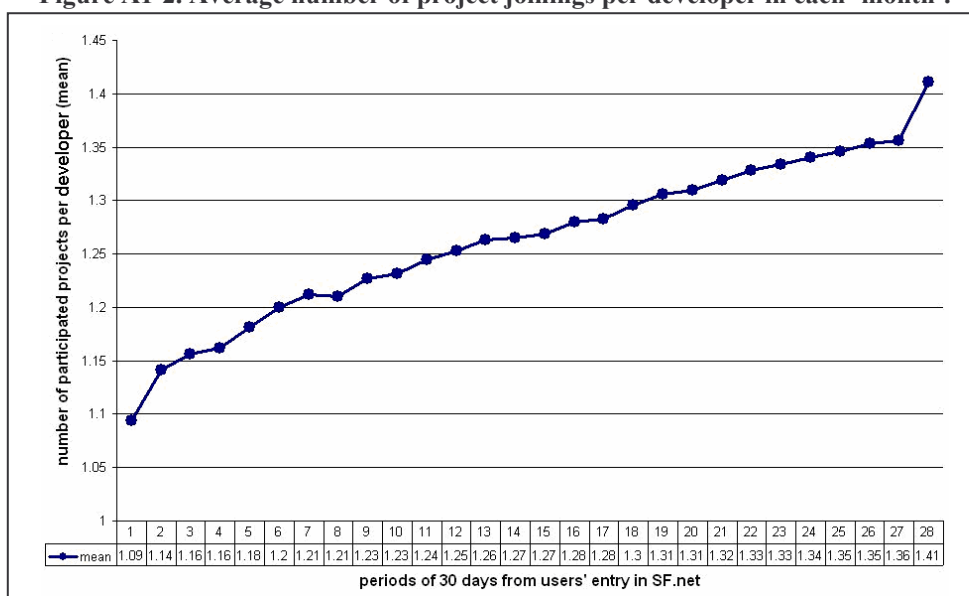
<i>N peak</i>	POLY1	<u>Rank</u>	<u>Days</u>	DS1	<u>Rank</u>	<u>days</u>
1	0.3	1 st	628	0.52	4 th	362
2	3.14	2 nd	60	3.14	1 st	60
3	1.69	3 rd	111	1.8	2 nd	105
4	1.16	4 th	162	1.15	3 rd	164

The spectral densities reveal that there are 4 main cycles for both (de-trended) time series, and that they are more or less congruent, as would be expected. The frequencies show that the structure of cycles is not very coherent, so that we have to use a certain degree of approximation in evaluating what time period is the best for our purpose.

Participating in the projects

A similar sub-sample and a similar analysis can be built in the case of project membership. In this case, also the first 60 days can be considered, being the problem of imported outside-born projects described above less relevant, as Figure A1-2 suggests.

Figure A1-2. Average number of project joinings per developer in each ‘month’.



Developers registered from September the 1st to October the 30th, 2000, conditional on being a project member in the first 30-day ‘month’ their registration on SF.net.

Applying the same procedure undertaken before, we compute the spectral densities of both the detrended series. The space of the two spectral densities and their peaks is drawn in Table A1-2.

Table A1-2. Cycles of Project-Joinings

<i>N peak</i>	POLY1	<u>Rank</u>	<u>days</u>	DS1	<u>Rank</u>	<u>days</u>
1	0.27	1 st	698	0	1 st	<i>whole series time span</i>
2	0.78	2 nd	242	0.71	4 th	265
3	1.94	3 rd	97	1.94	2 nd	97
4	2.68	4 th	70	2.66	3 rd	71
5	-	-	-	1.26	5 th	150

Longest waves are the most important ones, as expected: a new cycle of 240 days appears, and **DS1** reports a cycle spanning the whole series. Also cycles of close to 60 and 100 days are still in. Eventually, **DS1** bears another cycle of 150 days, which is missing in **POLY1**.³⁵

³⁵ When allowing for a flat trend, i.e. a constant, Dickey-Fuller test assures the stationarity both of **DS1** and **POLY1** at the 1% level. As a further check we undertook the same analysis after twice-differencing the original series, i.e. computing the spectral density of **DS2**, and omitting the first and the last periods. In both cases the results reported in

Remarks

Bearing in mind that the window to observe the registrants only spans the months from September 2000 to December 2002, i.e. 28 months, adopting a long period for the transition matrix, for example one year, will seriously limit this analysis. The longer the period we choose, however, the more the cycles we are able to account for in the transition matrix. Given this constraint, choosing time spans of 6 months seems to be the compromise best suited to the circumstances. The 180-day window for events on the basis of which the individuals' activity states can be determined should amply allow for influences of antecedent short cycles, given the fact that it embraces the 60-day cycle, and also the 90/110-day cycles, and even the 150/160-day cycles – with small biases with respect to their lengths. Thus, periods of 6 months seem to be adequate to generate a first order Markov process.³⁶

Appendix 2: Analysis of Transition Matrix Homogeneity

As the text of section 3 describes, our aim in establishing that the transition probability matrix describes an ergodic Markov process is to be able to compute the limiting distribution of developers among the activity states as a means of describing the effect of the persisting micro-dynamic forces at work, free from the influence of the initial distribution (i.e. the distribution observed at the close of period *A*). For the same descriptive purposes it is important to examine the stationarity of the transition matrix, or, more formally, to test the homogeneity of the transition matrix with respect to time within our 360-day interval of observation.

In general, the usual methodology for studying homogeneity involves the division of the whole sample upon which the estimate of *P* is based into a number of sub-samples. To carry out a homogeneity test, the sub-samples have to be formed according to the dimension of interest. If the division is obtained following the same units of analysis in different periods, a test of the time stationarity (stability) of the Markov process can be performed comparing the transition probability matrixes computed for successive time periods of observation. If not significant differences are found, one can conclude that the process that generated the entire sample of observations is time-invariant, and hence can be said to be homogenous with respect to the time dimension.

This methodology obviously can be implemented to test for homogeneity of the process with respect to specific exogenous characteristics of the agents, such as gender, age at initial date of observation, or initial skill level. In this Appendix we present only the results from the analysis of time homogeneity.

The time dimension: stationarity of the process

A direct consequence of the stationarity of the process is the stability of the transition matrix over all the periods in the observed time span. This means that the transition matrix P_{ab} , i.e. the matrix expressing developers' movements from the first 180 days of their experience in *SF.net* (remember we fixed the 90th day as the starting point, so that the first epoch (labeled "a") goes from day 90 to day 270) to the second 180-days epoch (**b**), has to be equal to the matrix P_{bc} , representing the movements from the second (**b**) to the third 180-day epoch (**c**).

In order to check whether $P_{ab} = P_{bc}$ the whole sample of observations can be divided in two periods: the first, spanning the first 360 days of developers' experience in *SF.net* (from day 90 to day 450) and the second including the days from 270 to 630. This has an effect also on the number of developers who can be considered to estimate the matrixes. Not all the developers, in fact, had an experience in *SF.net* long enough to allow for a one-year-and-a-half long observation of their movements across states. Thus, the sample had to be reduced to the developers registered in the first 240 days of activity of *SF.net*, from September the 1st, 2000, to 28th April, 2001. The maximum likelihood estimate of the transition probabilities from the first 180-days period to the second period of the same length, and from this period to the third were then performed considering 104,698 developers. The sub-sample, which is roughly half of the whole sample used in the previous section, is nevertheless wide enough to give reliable estimates.

the table are basically confirmed. When **DS1** is computed without extremes the relative importance of the different frequencies is basically unchanged, but the 265 and 150-day cycles are replaced by a 200-day cycle. In **DS2** the relative importance of the longest waves decreases, both with and without the extremes.

³⁶ This time span, however, clearly would not remove the possible influences of some seasonal events that have an annual periodicity, such as the major holidays. However, seasonal events will be distributed over all our sample, because the considered time dimension is relative to individuals' experience in *SF.net*, and every seasonal event will be placed in a different 'month' of each individual's experience according to her or his registration date on the platform.

The two matrixes (Tables A2-1 and A2-2) seem quite similar, but there are some notable differences. For example, comparing the transition probabilities from state 6, it is easy to see that P_{bc} assigns a greater probability to movements to state 5 and puts less probability mass on movements to state 3. As may be seen from Figure A2-1, those are the most salient points of disparity between the two matrixes.

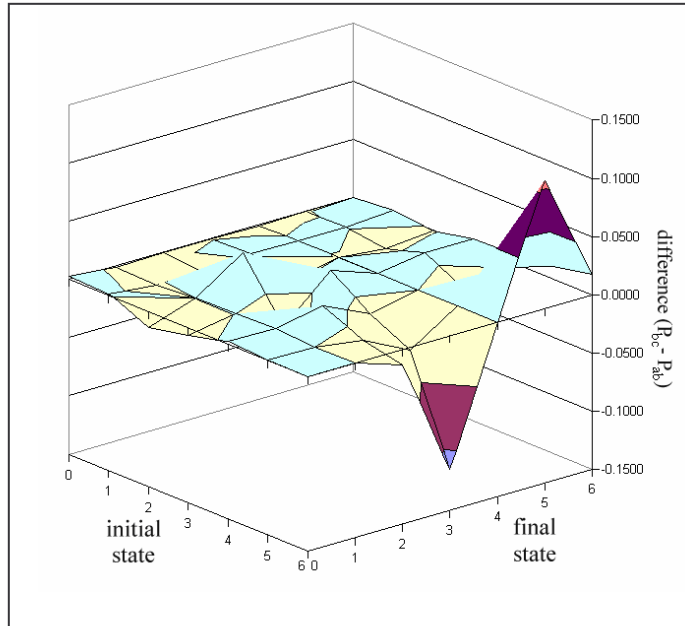
Table A2-1. The Transition Matrix P_{ab} .

States	0	1	2	3	4	5	6	# developers
0	0.9800	0.0081	0.0055	0.0005	0.0049	0.0006	0.0004	87766
1	0.7070	0.2190	0.0403	0.0019	0.0269	0.0048	0.0000	1041
2	0.0460	0.0010	0.8880	0.0318	0.0007	0.0285	0.0040	12031
3	0.0111	0.0000	0.0664	0.8562	0.0000	0.0622	0.0042	1898
4	0.0085	0.0000	0.8718	0.0502	0.0000	0.0641	0.0053	936
5	0.0037	0.0000	0.0386	0.8396	0.0000	0.1007	0.0174	804
6	0.0000	0.0000	0.0135	0.8288	0.0000	0.1081	0.0495	222

Table A2-2. The Transition Matrix P_{bc} .

States	0	1	2	3	4	5	6	# developers
0	0.9829	0.0070	0.0049	0.0004	0.0041	0.0003	0.0004	87335
1	0.7086	0.2159	0.0367	0.0031	0.0273	0.0052	0.0031	954
2	0.0329	0.0011	0.9157	0.0227	0.0006	0.0239	0.0031	12188
3	0.0051	0.0003	0.0446	0.8706	0.0000	0.0686	0.0108	2959
4	0.0108	0.0022	0.8812	0.0410	0.0000	0.0518	0.0130	463
5	0.0044	0.0000	0.0324	0.8132	0.0000	0.1221	0.0279	680
6	0.0000	0.0000	0.0000	0.7143	0.0000	0.2185	0.0672	119

Figure A2-1. The difference ($P_{bc} - P_{ab}$)



To statistically assess the degree of resemblance between P_{ab} and P_{bc} , we perform a test based on Bickenbach and Bode's (2001: p. 12) modified version of the test introduced by Anderson and Goodman (1957: p. 97).³⁷ The application of the Bickenbach and Bode's modified test is more appropriate where one or both of the transition matrixes have some zero entries, as is the case here.

We denote the actual transition probability from state i to state j by p_{ij} , and its estimate obtained through a maximum likelihood procedure by \hat{p}_{ij} . Anderson and Goodman (1957: p. 95) show that $\sqrt{n_i}(\hat{p}_{ij} - p_{ij})$ has a limiting

³⁷ See also Amemiya (1985: p. 417).

normal distribution with zero mean. Thus, as n_i , the number of developers in state i in the first period grows, we have that

$$\sum_{i=1}^N \sum_{j=1}^N n_i \frac{(\hat{p}_{ij} - \underline{p}_{ij})^2}{\underline{p}_{ij}} \sim \text{asy } \chi^2(N(N-1)) \quad (1)$$

where $i=1, \dots, N$ and $j=1, \dots, N$.

The same statistical property can be used to test whether the estimated transition probability p_{ij} significantly differs from a specified value p_{ij}^0 belonging to a non-random matrix P^0 . The test statistic is obtained simply by substituting p_{ij}^0 to \underline{p}_{ij} in equation (1):

$$\sum_{i=1}^N \sum_{j=1}^N n_i \frac{(\hat{p}_{ij} - p_{ij}^0)^2}{p_{ij}^0} \sim \text{asy } \chi^2(N(N-1)) \quad (2)$$

In the present case, we want to compare \mathbf{P}_{ab} to \mathbf{P}_{bc} , so that we can perform two different tests considering, alternatively, \mathbf{P}_{ab} or \mathbf{P}_{bc} as the matrix containing the non-random elements³⁸ p_{ij}^0 . Notice that the test has to account also for the possibility that $p_{ij}^0=0$ for some i and j . Following Bickenbach and Bode (2001: p. 12), we can adapt the previous statistics to our needs by excluding from (2) the summands for which $p_{ij}^0=0$ and correspondingly adjusting the degrees of freedom of the χ^2 . In particular, $N(N-1)$ has to be decomposed in order to take into account only non-zero elements in each row i .

Thus, when testing whether the estimate of \mathbf{P}_{ab} is equal to the elements of \mathbf{P}_{bc} , under the null hypothesis H_0 : $\hat{p}_{ij}^{ab} = p_{ij}^{bc}$ the statistic and its distribution are obtained as:

$$\sum_{i=1}^N \sum_{j=1}^{N_i^{bc}} n_i^{ab} \frac{(\hat{p}_{ij}^{ab} - p_{ij}^{bc})^2}{p_{ij}^{bc}} \sim \text{asy } \chi^2 \left(\sum_{i=1}^N (N_i^{bc} - 1) \right)$$

where N_i^{bc} is the number of non-zero elements of \mathbf{P}_{bc} on row i and n_i^{ab} the sum of their observations. Instead, if the null hypothesis is H_0 : $\hat{p}_{ij}^{bc} = p_{ij}^{ab}$ the test statistic becomes:

$$\sum_{i=1}^N \sum_{j=1}^{N_i^{ab}} n_i^{bc} \frac{(\hat{p}_{ij}^{bc} - p_{ij}^{ab})^2}{p_{ij}^{ab}} \sim \text{asy } \chi^2 \left(\sum_{i=1}^N (N_i^{ab} - 1) \right)$$

where N_i^{ab} is the number of non-zero elements of \mathbf{P}_{ab} on row i and n_i^{bc} the sum of their observations.

Considering \mathbf{P}_{bc} as the non-random matrix, the test-statistic is a χ^2 with 34 degrees of freedom, for which the critical values are 56.06 (for the 0.01 level of significance) and 44.90 (for the 0.10 significance level). As the computed test statistic in this case is 279.99, we can clearly reject H_0 .

A similar result is obtained when \mathbf{P}_{ab} is taken to be the non-random matrix. Here the critical values of χ^2 are found for 32 degrees of freedom at 53.49 (for 0.01 significance) and 42.58 (for 0.10). Again the value of the test statistic is high enough (244.34) to decisively reject H_0 .

Remarks

To conclude, the Markov process cannot be considered stationary. There is a significant evolution of the dynamics of agents' behavior on the *SF.net* platform, which may be described in the following terms. With the passage of time it become less likely that during a 6-month period developers who are members of multiple projects while launching more than one new project (i.e., are in state 6) will revert, in the following 6 months, to simply belonging to one project (state 3); instead, it is more likely than formerly that they will be found to have retained multiple project memberships and launched a new project. It is not possible on the basis of these observations alone to venture whether this change does or does not reflect a learned propensity on the part of agents for launching successive new projects.

³⁸ In the absence of randomness of both the estimated matrixes, the test reduces to an illustrative assessment of the difference. The distance between the values of the statistics and the critical values, however, assure a certain degree of robustness to this check.