Notes and Slides from a Presentation to the OSS Watch Conference on

Open Source and Sustainability

held at the Saïd Business School, Oxford 10-12, 2006

A Multi-dimensional View of the "Sustainability" of Free & Open Source Software Development

Sustaining Commitment, Innovation and Maintainability with Growth

By

Paul A. David
Oxford Internet Institute & Stanford University
paul.david@oii.ox.ac.uk & pad@stanford.edu

Revised version: 7 June 2006

Introduction: An economic systems perspective on the question of sustainability

Questions of "sustainability" are at the heart of much of the speculative discussions about the long-term significance of the open source software movement. Some of those questions concern what I will refer to as the characteristic community-based mode of developing *FLOSS* (Free/Libre and Open Source Software), associated with large projects such as Linux, Apache, GNOME, KDE, Mozilla, MySQL, and many others. That is not the entirety of the sustainability question, but it the subject I want to take up in this presentation: Is the FLOSS mode of production really a lot more substantial than the name would suggest?

Much of the discussion of sustainability has been cast exclusively in terms of the need to provide funding by coupling projects with the direct or indirect sponsorship of business firms that can expect to profit by marketing complementary goods and services, from hardware and grid services to software manuals and IT consulting. By pitching the question of sustainability at the level of the individual project, and deliberately pressing for a "hard-headed" view of "financial" matters that do not figure centrally in the organization and conduct of FLOSS projects, that approach certainly introduces a useful note of economic realism into discussions of "the future of open source."

Yet, it must also be said that to consider the issue of sustainability solely in those terms tends to have the undesirable effect of presenting the problem and its solution as an "add on activity," albeit one that wise project leaders should be thinking about in advance in order for support arrangements to be in place as their development effort matures. To disconnect one's thinking about sustainability issues from the organization and conduct of the development process in that way, however, is likely limit the effectiveness of the advice at the level of individual projects. Moreover, it takes attention away from a number of significant issues that are likely to affect the collective future of FLOSS as a mode of software production, and possibly as a paradigm for information goods production more generally.

An alternative (but not incompatible) way of looking at a number of other facets, or dimensions the sustainability problem is provided by taking the "economic systems" approach that I will pursue in this brief presentation. Will the "distributed community mode" of software

SOME RIGHTS RESERVED This work is licensed under the Creative Commons Attribution-NonCommercial-NoDerivs 2.5 License. To view a copy of this license, visit http://creativecommons.org/licenses/by-nc-nd/2.5/ or send a letter to Creative Commons, 543 Howard Street, 5th Floor, San Francisco, California, 94105, USA.

development – which is a distinctive generic feature, shared by the variety of different specific FLOSS project "models" – be sustainable? Can this "system" not only continue producing robust code, but generate novel, innovative software, while maintaining and upgrading the code of the projects that have already established themselves as challengers to the conventional commercial sector of the software industry -- not only in the web-server market (Apache), but in field of browsers (Mozilla-Firefox), text processing (Open Office), relational databases (MySQL), and most recently, multi-player on-line games? In other words, do organizations of this kind – being neither markets or hierarchically managed firms – have a future, so that it will benefit other private and public agents in the economy and society to think about how best to interact with their members and their products?

We can begin to tackle this big question by focusing on three more manageable facets of the problem of sustainability: "community commitment," the rate of innovation through the founding of new projects, and managing maintainability in large projects as they add functionality. I do not claim to have definitive answers on these topics, much less to have reached a conclusion about the big question. The message I wish to convey is one that is mixed: what empirical research on FLOSS developers and their projects has shown is that at least some of the doubts that continue to be voiced about the future of this movement are not well supported by the evidence. Still, the outlook is far from perfectly rosy, because there remain a number of less frequently noticed challenges to "system survival" – which from my perspective would be "success."

Sustaining commitment in the community of FLOSS developers

Ideological enthusiasm undoubtedly plays a role in the movement's recruitment of volunteers, but isn't that bound to dissipate when those who have joined come to face the realities of earning a living? Won't those who were most strongly attracted by the non-commercial "communitarian" ethos of FLOSS soon become disaffected by either the economic rewards reaped by a few successful charismatic project leaders, or by the growing corporate employment of expert professional programmers to work on the projects?

Why not look at the evidence from developer survey responses? What are the main motivations for beginning to participate in FLOSS development activities, and how do these evolve with experience? Do developers tend to become less ideologically and community motivated?

- It's a mistake to imagine we will find a typical or representative "motive" to explain the behavior of developers that are involved in FLOSS activities. Many motives are offered by survey respondents, and it is usual for individuals to give multiple reasons as having been important in motivating them to begin contributing to FLOSS projects. The reasons range from the ideological to the material; from the "intrinsic" satisfactions programming and bug-fixing to the instrumental goals of increasing one's software skills and creating code to meet a special software need; and from individualistic ego-driven motivations to more other-regarding, community-oriented rationales for action.

- But individuals' motives are not static. In the FLOSS-EU (2002) survey of developers, the reasons respondents give for their continuing participation are more focused than the reasons that motivated then to begin: among the focused motivation-profiles identified by cluster analysis, the two that emerge as most prominent can be characterized as "ideological and community-oriented", and "individualistically instrumental" (e.g. improving their programming skills and producing software that would be useful to them).

- So, the importance the developers attach to their personal identification with community norms and ideology actually doesn't diminish as they gain experience in FLOSS development (at least not among those who do continue participating); and for many developers their social motivations are entirely compatible with assigning importance the practical benefits they derive as individuals.

- Nonetheless, among the developers who responded to the 2003 FLOSS-US web-survey and, equally, among those who registered and were active on *SourceForge* in the same period, the typical early experience is participation in very small, and little known projects – rather than attachment to large communities. Perhaps this is less the case among developers in the EU than those in the U.S. and other regions, but *SourceForge* is thought to be a very internationally varied site for FLOSS projects, and 72 percent of the projects listed on at the close of 2003 had only 1 member!

From the organizational perspective, however, it is the large and complex development projects that constitute the interesting and novel departure in software production, making the strength of individual motives relating to community identification and social interaction a critical matter for the sustainability of "community commitment." Are the motives and attitudes of developers participating in large FLOSS projects significantly different in those respects from the motives of the individuals who are devoting their efforts to one or another of the myriad small projects?

- A recent analysis of developer motivation patterns (based on the FLOSS-US 2003 Survey data) defined clusters, two of which were distinguished by the strength of the relative importance the individuals attached to social interactions in code development or in acquiring software knowledge and skills --along with other reasons for their participation that were shared by individuals assigned to other "clusters."

- Comparing the relative frequencies of those two motivation-clusters among the developers who could be identified as contributors to large (>29 member) or to very small (>1-2 member) projects, it is found that the projects in the larger size range had attracted a significantly larger share of the developers who were motivated by those more "socially oriented" forms of participation in FLOSS. The evidence of the existence of such an association is perhaps not so surprising, but one still cannot say in which direction the causation runs between project size and the intensity of the value placed by its contributors on social interactions within the community of developers.

Does this mean that, by-and-large, FLOSS developers are un-sympathetic to the attractions of business applications and professional career advancement in software production based on open source, or doubtful that they would be able to earn a living in this way?

- Clearly not, because a majority of the respondents to the FLOSS-US (2003) say they expect to have future roles in open source-based businesses. About 66% of those in the age group 23-28 answered positively, indicating they looked forward to being either owners (15%) or employees (26%) of such companies, or consultants (22%). Among the 28-34 year-olds, the corresponding percentages are still larger: owners (25%), employees (43%), and consultants (38%), so that 95% of that age-cohort anticipate earning some part of their living in this sector of the software industry.

- Such expectations are somewhat more prevalent among the survey respondents who report having begun their participation in FLOSS activities during 2001-2003. We cannot say whether this reflects the emergence of a business ecology organized around the growing number of mature FLOSS products, or whether that trend itself is being driven by more pronounced business interests of those who are being drawn into "the world of FLOSS."

- At the same time, as several survey-based studies have confirmed, *immediate* economic rewards in the form of "direct monetary earnings" are not at all central in drawing the mass of developers into contributing to FLOSS projects, whatever the expectations are about the future. Furthermore, as I have pointed out earlier in noting that motives evolve on the basis of experience, where reasons for continuing to participate can be distinguished from those for

starting, it is the more ideological and community-oriented reasons that are found be salient for a larger proportion of survey respondents.

It surely is a mistake to think of monetary rewards and other non-monetary considerations as alternative, mutually incompatible sets of "inducements" or incentives for working on FLOSS projects, or, indeed, in any line of work. Although the theory of "compensating differences in pay" goes back to Adam Smith, and there is ample empirical evidence that people ask for higher wages to do less pleasant, or more dangerous work, it is also true that individual "values" of a non-materialist kind and preferences for association with the goals of some employing organizations rather than others, affect the ability to recruit workers for a given task. Nonetheless, it is relevant to try to answer the question: Which is the more important -- work that direct earns monetary compensation or pure volunteer work – as a source the current developers' efforts devoted to the production of open source software?

- Monetary rewards undoubtedly are a large part of the motivational picture, but this is not the same thing as saying that immediate direct income earning (as opposed to hopes of future and possibly indirect economic payoffs) is the main consideration sustaining participating in FLOSS activities. On the basis of the two major surveys of developers --FLOSS-EU (2002) and FLOSS-US (2003) – which between them obtained almost 4000 responses to the same set of questions about monetary earnings from open source software work, the bounds on the proportion of developers who report receiving direct compensation can be put at 27.6% -- 33.1%. The others (i.e., the 68% to 72%) who may be viewed for this purpose as the pure "volunteers", reported either no monetary earnings of any kind or only "indirect income" earned in a job for which they had been hired the basis of skills and experience gained previously through involvement in FLOSS development.

- Taking the upper bound of 33% as the non-volunteer portion of the FLOSS developer community, one must then make an allowance for the difference in the relative time devoted to open source projects by those who are employed to do it, and those who volunteer. Both the FLOSS-EU and FLOSS-US survey's concur in reporting an average 11 hours per week worked on open source projects, but if one assumes that the "volunteers" put in only half as much time per week those who were receiving direct monetary compensation, the two groups would be contributing equally in terms of time input. Assuming that those who are compensated work twice as much as the "volunteers", on average, is not entirely arbitrary: The FLOSS Survey asked the developers who were employed to work on proprietary software how much time they spent in such work, and found the weekly average hours were roughly twice the general average spent on open source development. (But it should be noted that this doesn't give us exactly the multiplier that we need for the calculation. In the first place, only one-fourth of the "non-volunteers" were employees whose assignment was to work on open source projects, so there is a question as to whether the self-employed averaged longer or shorter hours than those who where employed. A second problem is that employees of proprietary software firms -- most of whom are unlikely to have been assigned to work exclusively on open source projects -- would be constrained in the amount of time they could devote to open source development, and that could would make the multiplier of 2 rather too high.)

- But its not a major worry if the time input multiple of 2.0 and hence the 50% effort estimate for the non-volunteers is overstated, because it seems reasonable to assume that the average effectiveness of an hour worked on open source development by a paid employee would exceed that of a hour of effort from the average "volunteer." To go farther in assuring that this exercise is not overstating the relative contribution of the "volunteers" in terms of their aggregate equivalently effective effort, one could allow that paid employees might be 25 % more effective on average. The upshot is that two-thirds of the FLOSS developer community that are "pure volunteers" contributed about 45% of the "equivalent effort input, and the

balance being supplied by those who receive at least some form of direct monetary compensation for their FLOSS development work.

- It should be evident from the foregoing discussion that the role of non-monetary "rewards", expectations of the future viability of income-earning careers based on open source software, in mobilizing the effort of volunteers, along with the proficiency of recruits and the support provided for skills improvement, are not issues that can be neglected in discussions of the long term sustainability of the open source mode of production any more than they can be taken for granted by those who hope to lead large successful projects.

Sustaining the generation (founding) of new FLOSS projects

With many of today's FLOSS developers seeking to become better programmers, and expecting to find future employments in businesses based on open source software, does that translate into a sustained rate of creation of new projects? Is there any reason to suppose that the proportion of innovating, entrepreneurial types in the population of FLOSS developers will remain as high as it has been until now?

- Actually, if launching one's own project is the test of having the "entrepreneurial spirit," the trend seems to be going in that direction, rather than the other way. Among the respondents to the FLOSS-US (2003) survey of developers, the proportion of responders who were working on projects that they said they had launched alone or with others is strikingly high: 52 percent said their current project not well known and that they had founded it, either alone or with others.

- Moreover, reports of that kind are especially frequent from those who had taken up FLOSS most recently (during 2001-03) and were working on their first and only project: 82 percent described the project as "unknown" or "known only slightly" and as one that they had had a hand in its launching.

- Of course, as is the case with small business start-ups, the odds are that the mass of these little known projects will remain that way and will eventually be abandoned, whether they achieve "maturity" or not. The question here is about the propensity to attempt to succeed by creating something new, not the probability of succeeding in an innovative project.

  -- *Creating code to form the kernel of a new project is harder than releasing existing code under open source license. So rising numbers of projects on platforms like SourceForge could be quite misleading. Existing code brought to the platform and released as open source might not have the architecture suitable to absorb contributions from those who hadn't (re-) launched the project, whereas projects created de novo in the environment of SourceForge could have a better chance of attracting a growing community of contributors. But aren't many of the projects that appear on SourceForge attracted to that site, rather than really being "born" there?*

- Yes, that is true, especially during the early years of *SourceForge*'s history: during the first 3 years, i.e., up though 2001, the proportion of registrants who posted projects during their first three months on the platform was higher than it subsequently became, as it declined with each successive (60-day long) cohort of entrants; a similar cohort-to-cohort decline cut the proportion of projects posted during months 4 through 9 to half of its initial level.

- On the other hand, since most of the registrants on *SourceForge* (more than 80 percent) are just "sightseers" who did not join a group, submit a patch or even contribute to a project

forum, it is more informative to focus on the project-launching propensity of those belonging to the "active" sub-population: among the latter, the proportion that became founders during months 4 through 9 on the platform remained little changed from one-cohort to the next.

- As the average probability that an "active" developer will launch at least one project within a 6-month interval is approximately 0.12, we can expect at this rate that within a two-year period about one-fourth the members of an active developer cohort would have joined the ranks of founders, and three-fourths of them would do so within a five-year interval.

- Even though the "active" developers on *SourceForge* constituted less than 15 percent of those who registered themselves (and were not de-listed), the propensity to launch a project, as just reported, is quite high. But it is in substantial part a transient condition, reflecting the likelihood that developers who are going to launch a project on the platform typically do not wait many months before doing so, and that thereafter they tend to become absorbed with that the activities of that venture and do not launch another.

- Therefore, in the limit – that is to say, abstraction abstracting from the transient influence of what they were doing (or not doing) during the months immediately after entering *SourceForge* -- the proportion of active developers that can be expected to launch just one project within a 6-month interval is about one-third lower than the empirically observed rate, or 0.08. This average probability implies that within 5 years 57 percent of them (rather than 75 percent) would have founded a new project.

- Evidence from the 2003 FLOSS-US survey is quite consistent with the foregoing picture. The median duration of the respondents' participation in open source activities is 5 years, and 47 percent of them reported having "launched" their current project, either alone or with others. Among those who had only start to participate during 2001-03, however, 42 percent reported having launched a project. Given the shorter time interval on which the latter group are reporting, the implication is that (in the sub-population whose experience was contemporaneous with the developers observed on *SourceForge*) the proportion founding a project within a 5- year interval would be higher than 47 percent. Therefore it would more closely approach the 57 percent estimate obtained from the study of active cohorts on *SourceForge*.

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 are just beginning, isn't it rather unlikely that the skills and experience of typical "founders" differ very markedly from the software programming skill of the developers individuals who prefer to join existing, larger project?.

- This supposition is confirmed by the close similarity between the average self-reported level of software skills in the sub-population who had launched a project on *SourceForge* and that among SourceForge project members who never had a founder's role.

- The founders of projects do differ, on average, in other regards: notably in the greater frequency of their generic skills and communication and networking experience, whereas they typically have had less experience as project member, particularly on the projects with many members.

- There still other, rather intriguing indications that the propensity to join projects does not fit well with being a founder of projects. It is possible to compare the activity records of two groups of SourceForge participants, both of observed to have become project members during their third month on the platform: for those who joined a single group at that time, the limiting expected proportion that would launch a project within a 5-year interval was slightly

greater than 26 percent; whereas among those who initially had joined more than one group, the corresponding proportion of founders was negligibly small.

Sustaining maintainability as functionality is added to the large FLOSS projects

The world of large FLOSS projects – those that have attracted the widest notice and are arguably are the most important – is a very different one from that of small, personally managed projects. In general, the lives of small projects, like the lives of small business ventures, are more precarious and their survival is more difficult to predict. But is the continuing growth and maintainability of large projects FLOSS projects much more certain?

Quite aside from the issues of motivation and commitment of developers associated with these projects, doubts have been expressed concerning the validity of drawing an analogy with the greater survival capabilities of large firms. Whereas the latter typically are guided by the "visible hands" of a hierarchical management structure, much less is known about the governance of large FLOSS projects.

Moreover, the literature on software engineering contains a strand of argument which leads to the expectation that these projects will become more and more difficult to maintain, and that their pace of growth (and increasing functionality) will slow.

Indeed, the signs of "crisis" that overtook the development of Microsoft's promised "Longhorn" update of Windows, and the more widely reported delays experienced with the successor ("Vista") project, have suggested to some commentators that conventional commercial software may be no less vulnerable to such problems.

- The problems encountered by the once-vaunted development process at Microsoft are something of a red herring in this connection. There are at least two grounds on which to dismiss the experience of Windows as largely irrelevant:
    - In the first place, the Windows code base is far bigger than even the largest FLOSS projects, e.g., Linux.
    - Secondly, and perhaps more importantly, in the case of Windows strategic business reasons led Microsoft's programmers to embedded new functionalities (browser, media player, etc.) in the operating system, greatly complicating the task of updating the resulting complex architecture of the system.

Leaving the experience of the commercial software development organizations aside, then, , what does the available more direct evidence suggest regarding the existence of maintainability problems that create obstacles to the growth large FLOSS projects?

- Remarkably, it is found that the growth of the file-size of the Linux kernel has been super-linear, and in a sample of 17 large projects, the time path of project file-size is substantially linear.
- There are two conditions that suggest that a constant rate of growth, involving the addition of functionalities, can be sustain in these large FLOSS projects—at least over the size range bounded by the Linux kernel project:
    - The ability of the projects to mobilize a rising number of distributed developers and bug-patchers distinguishes their situation from that of the commercial software companies, and therefore if growing complexity brings increased need for programming inputs, it need not slow the development process. Instead, it is likely to call for increased number of maintainers to deal with code submissions and handle

commits: It is relevant that a recent study of file level activity in 10 large *libre* software projects has found that there is a systematic positive association exists between technical measures of code complexity (specifically, either McCabe or Halsted complexity) and the maximum number of maintainers that commit to the file.

- Large FLOSS projects are not only modular in their architectures –in the sense that the code is organized in a large and increasing number of "packages" or module (as illustrated by the Linux kernel releases 1.0, 2.0 and 2.5), but the architecture can be designed so that cross-dependence among the modules is minimized, making the organization of sub-projects more separable, and hence more readily governed by the maintainers.

Therefore, on technical grounds it appears there are no obvious and compelling reasons to think that large FLOSS projects will share the particular fate that has overtaken the development of Windows, so long as they remain able to mobilize and elastic supply of competent and committed programmers. Of course, that is not assured and it is one of the grounds for realistic concern about the long-term future of this mode – or indeed any mode -- of software production.

A realistic optimist's conclusion

Correcting a number of frequently voiced grounds for pessimism doesn't dismiss all the skeptics questions, because there remain are a number of serious challenges to the sustained success of the FLOSS community mode of production – and that is the only viable alternative to conventional "closed" code production methods for complex software systems.

A number of the potentially more serious challenges have been alluded to in the preceding discussion, but here is a more explicit and extensive list:

- Will there be an adequately elastic supply of FLOSS programmers who are capable of writing robustly reliable code? Are the many projects that have only a few members indicative of difficulties in recruiting a talented core of developers?

- Modular architectures are essential for successful management of the distributed development process, and to make it feasible for newly recruited programmers on big projects to quickly grasp the underlying reasons for the decisions that were previously made about requirements and code design. But it appears that the necessary talent for that challenging form of creativity -- designing an architecture that effectively anticipates and provides options for development of future functionalities, while not overly-specifying many other aspects of the future code – is quite rare. How can it best be identified and allocated among the communities that are developing large and complex system?

- The global FLOSS community is widely distributed and very loosely connected, so that coordination on some strategies of resource allocation is difficult. Will this be challenge addressed, or will the collectivity of major FLOSS projects essentially remain guided by emulative strategies of engaging focused commercial vendors of established software systems in head-to-head competition? This could put them at a disadvantage, especially when they seek to compete in niches where incumbent firms with large user bases have ample resources to move toward emulating FLOSS production methods, by taking up "agile programming" and moving toward more continuous release policies.

- The success of a number of large projects could make them targets for future patent infringement suits (like that of the SCO Group), pushing them toward protective limited

liability incorporation as foundations, but the resulting controls are likely to conflict with the ethos of the original FLOSS movement.

- The 'hacker ethos' that remains alive within many of the FLOSS communities (understandably) attaches highest status to creative programming, but the social and economic impact of the software will depend ultimately upon its adoption by large sections of the general population who are have little or no programming skills. To reach those "users" requires mobilizing expertise and sheer effort in a range of activities that presently are not accorded high status: designing user-friendly interfaces, linguistic (which is to say "cultural") translation, effective preparation of individual user manuals and guides to managing effective "organization migration" from closed software packages. This is an entirely different approach from that of leaving the "technology transfer/marketing tasks" to be undertaken by commercial enterprises --which typically are not closely integrated into the project's community, and which tend to wait until the project reaches "maturity" in order to see whether it is likely to attract a substantial "final user-base."

- …and, for still other grounds for concern, read Brian Fitzgerald's stimulating essay: "Has Open Source a Future?" (2005).

To deal with these challenges will not automatically solve the problems of providing adequate commercial sources of funding for core development and continued maintenance of major FLOSS projects. Nor could it guarantee willingness on the part of public agencies and charitable foundations to provide continuing overhead support for maintenance of mature project code.

Although I cannot say that dealing successfully with these difficult problems would be sufficient for success, finding effective ways to address many of them will be required for the sustained viability of the FLOSS paradigm as an alternative to closed commercial software production.

Acknowledgements

The research on which this presentation has drawn received support under grant awards to Stanford University from the National Science Foundation: IIS-0112962 (2001-04) and IIS-0329259 (2003-05). [See http://siepr.stanford.edu/programs/OpenSoftware_David/OS_Project_Funded_Announcmt.htm.] It rests also on the work of members of the international Project for Research on the Economics of Free/Libre & Open Source Software (*PREFLOSS*), which has been sustained by funding under those NSF awards by local institutional support from the Stanford Institute for Economic Policy Research (SIEPR) and its European academic partners: MERIT (at Rijksuniverseitet Maastricht), IMRI (at Université Paris-Dauphine), Informatics-GSyC (at Universidad Rey Juan Carlos, Mostoles) , SPRU/INK (at Unversity of Sussex); as well by as the Oxford Internet Institute' support of the Oxford Workshop on Libre Software (OWLS-2004) and Project *CALIBRE*-- an EU FP6 Coordination Action.

Selected References: sources and citations in accompanying presentation slides

Dalle, J.-M. and P.A. David (2005), "It takes all kinds: simulation modeling perspective on motivation and coordination in *libre* software development projects," SIEPR Open Source Economics Project Working Paper, Stanford University (October). Submitted to *Management Science*.

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 *how Open Will the Future Be? Social and Cultural Scenarios based on Open Standards and Open-Source Software*, eds. M. Wynants and J. Cornelis, Brussels: VUB Press, 2005.

Dalle, J.-M., M. den Besten, "Maintainer activity dynamics on large libre software projects," EC CALIBRE Project Working Paper, University of Paris-Pierre et Marie Curie, (April). [To be presented at International Conference on Open Source Software- OSS2006, in Como, Italy, 10-12 June 2006].

P. A. David, J.-M. Dalle, R.A. Ghosh and F. Wolak (2004), "Open source development and the 'economy of regard': evidence from code-signing behaviors in the Linux kernel," SIEPR Open Source Economics Project Working Paper, Stanford University. (December). Paper presented at the Conference on the Economics of the Internet and Software, University of Toulouse, January 2005, a revision of the paper for the OWLS Workshop, Oxford Internet Institute [See http://www.oii.ox.ac.uk/fiveowlsgohoot/postevent/David_OWLS_slides-blk.pdf.]

David, P.A and F. M. Rullani (2006), "Lurking, laboring and launching on SourceForge: Micro-dynamics of open source software development," SIEPR Open Source Economics Project Working Paper, Stanford University (March). [To be presented at International Conference on Open Source Software- OSS2006, in Como, Italy, 10-12 June 2006].

David, P.A., J. Shapiro and A.H.Waterman (2006), "What do we know about the developers who contribute to "community mode" production of *libre* software?" SIEPR Open Source Economics Project Working Paper, Stanford University (March).

David, P.A., A. H. Waterman and S. Arora (2003), "FLOSS-US: The free/libre/open source survey for 2003: First Report," SIEPR Open Source Economics Project Working Paper, Stanford University (September) [see http://www.stanford.edu/group/floss-us/report/FLOSS-US-Report.pdf].

Fitzgerald, B. (2005), "Has Open Source a Future?" Ch. 5 in *Perspectives on Open Source Software*, eds., J. Feller, B. Fitzgerald, S. A. Hissam and K. R. Kakhani. Cambridge, MA: MIT Press, (2005).

Ghosh, R. A., R. Glott, B. Krieger and G. Robles (2002), "Survey of developers (Free/libre and open source survey and study)," Technical Report, International Institute of Infonomics, University of Maastricht, The Netherlands (June). [Available at: www.infonomics.nl/FLOSS/report].

Glott, R., R. A. Ghosh and B. Krieger (2004), "Motivations of free/libre and open source developers," International Infonomics Institute Working Paper, University of Maastricht (May). See the presentation by R. Glott, "Towards integration of research approaches to FLOSS communities," Oxford Workshop on Libre Software (OWLS), Oxford Internet Institute, June 25-26, 2004. [Available at: www.oii.ox.ac.uk/fiveowlsgohoot/postevent/owls-presentation_r.glott.pdf .]

Giuri P., M. Ploner , F. Rullani F., S. Torrisi (2004), "Skills and Openness of OSS Projects: Implications for Performance", LEM Working Paper Series 2004/19, Scuola Sant Anna, Pisa. [Available at:http://www.lem.sssup.it/WPLem/files/2004-19.pdf"].

Robles, G., J. M. Gonzales-Barahona and I. Herraiz (2005), "Evolution and growth of large libre software projects," In *Proceedings of the International Workshop on Software Evolution*, Lisbon: Portugal (September).

A Multi-dimensional View of "Sustainability" in Free & Open Source Software:

Commitment, Innovation, Maintenance and Growth

Paul A. David
Oxford Internet Institute & Stanford University
paul.david@oii.ox.ac.uk

A presentation to the OSS Watch Conference on
Open Source and Sustainability
held at the Said Business School, Oxford
10th -12th April 2006

INTRODUCTION

—An Economics Perspective on FLOSS

What is "open source" software?

- Free/Libre and Open Source Software (F/LOSS, or FLOSS) is distributed with the source code fully disclosed, and made available for use and modification without monetary charges.
- F/LOSS is software that is distributed under the terms of a special class of copyright licenses, primarily under the GNU General Public License (GNU GPL).

Reference: *The Open Source Definition* – version 1.9, available at: www.opensource.org/osd.html

(Originally written by Bruce Perren, this is now maintained by the Open Source Initiative.)

F/LOSS LICENSE CONDITIONS—IN BRIEF

"Free Software Licences" (FSF definition)

- **must include source code**
- **must allow distribution in source code as well as in compiled (machine code) form**
- **cannot restrict any party from distributing the software as a component of and aggregate distribution containing code from different sources**
- **must be "non-discriminatory" among licensees**
- **must ensure that programmers receive credit (blame) for their work**

"Copyleft" Software Licenses (FSF definition)

Authors who copy and distribute programs based on GPL'd code (derivative works) must do so under the GNU GPL license-- which contains the above provisions, plus this one.

What is so very interesting for an economist about the FLOSS development process ?

- Collective, distributed mode of creating (producing) an information-good: software
- Extensive voluntary participation by communities of skilled and neophyte software developers
- Novel use of IPR to distribute/publish software under "public domain-like" conditions
- Essential dependence of the production mode upon the "anti-proprietary" distribution regime
- Critical role of computer-mediated communications (CMC) for this production system
- Self-documenting nature of the process permits microlevel quantitative research on the organization of 'collective invention'

Four fundamental economic research questions about the FLOSS *Production Mode*

***At the Micro-level*:**

- **How are the human resource inputs mobilized?**
- **What kinds of inputs are supplied by participants in large project communities (*C-Mode* production – rather than *I-Mode*)?**
- **What factors motivate participants to devote effort to a particular project, and project task, rather than to other activities?**
- **How are these inputs allocated and coordinated within projects? (I.e., among tasks of a particular kind ,esp. coding, bug-fixing)**

Four fundamental economic research questions about the FLOSS *Production Mode*

***At the Meso-level*:**

- **How are resource inputs allocated among projects of various types?**
- **What mechanisms are used to govern projects, maintain code quality?**
- **How well does the system match software output to the needs of diverse users?**
- **What are the comparative costs in terms of resource use of FLOSS *C-Mode* production vs. closed-proprietary software?**

Starting from this perspective this presentation focuses on three (non-funding) dimensions of "sustainability"of FLOSS software development:

- **Sustaining commitment in FLOSS projects' development communities**
- **Sustaining rate of founding of new projects**
- **Sustaining maintainability as functionality (and size) grows in established projects**

Sustaining commitment in FLOSS projects' development communities

Is the initial enthusiasm and ideological commitment to "the movement" bound to dissipate?

Will those who were attracted to the movement begin seeking material rewards, and become disaffected by the few who are able "cash in" on their reputations as great project leaders?

Let's look at what the survey evidence can tell us about motivations among FLOSS developers.

FLOSS-US

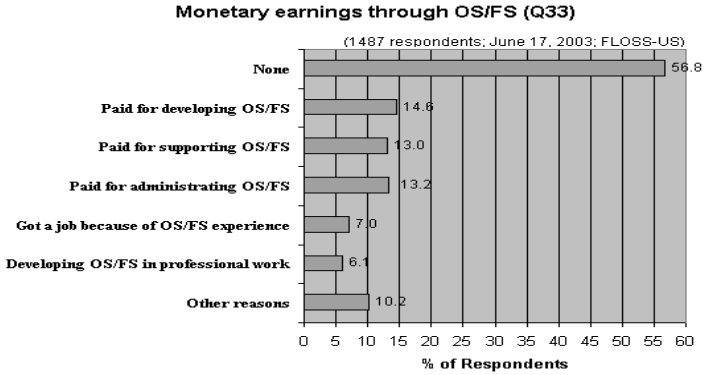
The Free/Libre/ Open Source Software Survey for 2003
To go immediately to the questionnaire, click here:

A Web Survey of Software Developers
conducted by the Stanford University (SIEPR) research project on

Economic Organization and Viability of Open Source Software
With funding support from the National Science Foundation.
http://siepr.stanford.edu/programs/OpenSoftware_David/OS_Project_Funded_Announcmt.htm

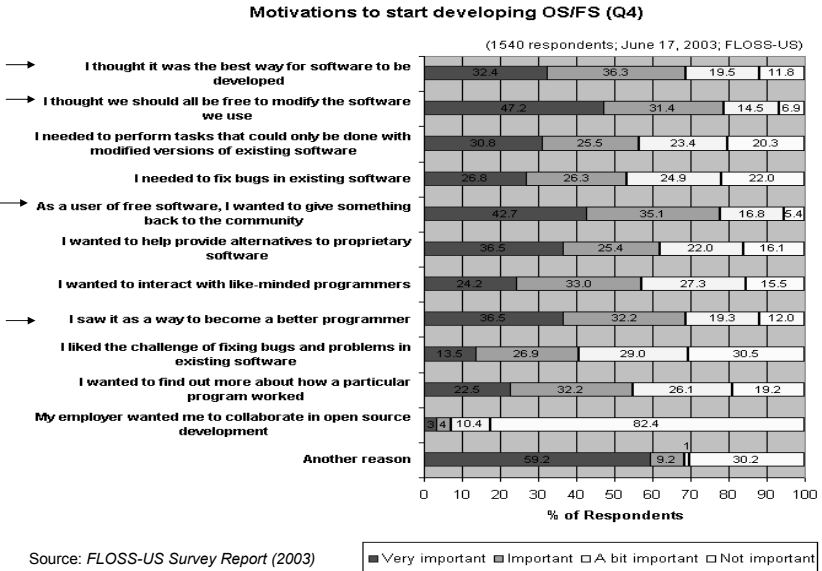
The FLOSS-US Survey: First Report
(September 2003) is available at:
www.stanford.edu/group/floss-us/report/FLOSS-US-Report.pdf

Income earnings from FLOSS activities remains atypical among survey respondents: 57% of FLOSS-US respondents cite not having direct or derived earnings benefits from their activities.



Source: *FLOSS-US Survey Report (2003)*

Ideological and self-improvement motives are salient among initial motivations of FLOSS-US (2003) developers



Source: *FLOSS-US Survey Report (2003)*

Motivations are not static -- they undergo change, and the question is whether they change in ways that are likely to allow them to be fulfilled by the experience of participating in FLOSS activities:

Initial Motivational Groups	n	%
Enthusiasts	**104**	**3.7**
Software improvers	**285**	**10.2**
Recognition seekers	**308**	**11.1**
Materialists	**327**	**11.7**
Ideologists	**472**	**17.0**
'Triers' (= diffuse motives)	**1288**	**46.3**
Total	**2784**	**100.0**

© 2004 International Institute of Infonomics / Merit
Source: R. Glott et al., Motivations of Free/Libre and Open Source Developers. May 2004.

This is the distribution of FLOSS-EU Survey (2002) respondents among the main motivational groupings identified by principal components analysis of their reasons for initially participating in open source software development….

…and this is they way motivation profiles of the same developers appear to evolve: stated reasons for continuing to work on FLOSS projects do not shift towards pursuit of economic rewards

Flows from initial to continuing motivational groups within the FLOSS community

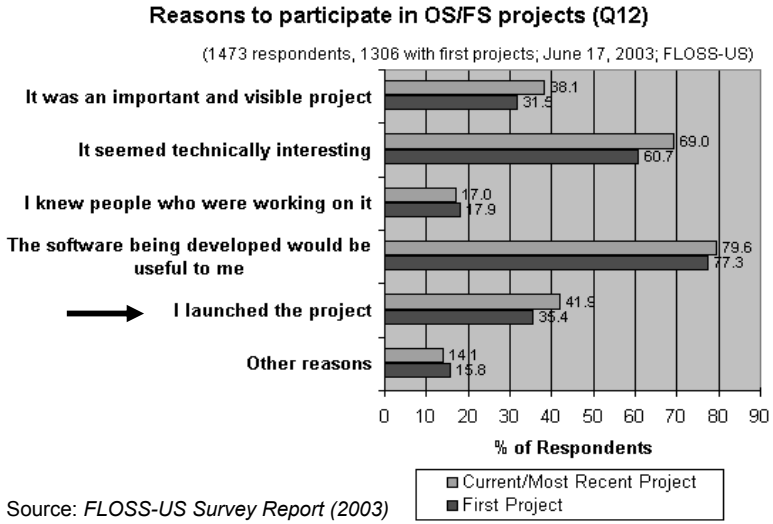
CONTINUING MOTIVATIONAL GROUPS	INITIAL MOTIVATIONAL GROUPS: enthusiasts	software improvers	recognition seekers	materialists	ideologists	'triers'	total
recognition seekers	33.7	8.0	41.6	14.1	7.2	6.8	12.0
skills improvers	32.7	22.8	37.0	26.3	43.6	31.1	32.5
software improvers	30.6	37.5	13.0	39.4	14.4	23.2	24.2
ideologists	2.9	33.7	8.4	20.2	34.7	39.9	31.2
total	100	100	100	100	100	100	100

Figures are percentages of the initial motivational groups. © 2004 International Institute of Infonomics / Merit

Source: R. Glott, R. A. Ghosh, B.L. Krieger, Motivations of Free/Libere and Open Source Software Developers, May 2004

The big group that cited diffuse motives ('triers') for beginning FLOSS have redistributed themselves toward "skill-improvement" and "ideological" reasons for continuing – motivation clusters that represent 64% of total respondents.

terms more pragmatic than the reasons they gave for participating in FLOSS development activities



Source: *FLOSS-US Survey Report (2003)*

The importance of "the personal utility of the software" among the reasons given by FLOSS-US (2003) respondents for their current project choices reflects the predominance of very small, *I-mode* projects that the respondents had personally launched.

- Of 1473 respondents listing a "current project", 64.8% described it as "unknown" or "slightly known': 33.0% launched it alone;
  : 46.8% launched it with others.
- Of 1306 respondents listing their "first projects", 61.7% described it as "unknown" or "slightly known"; 35.4% launched it with others.
- **Of 238 "newbies" (those starting a "first &current" project in 2001-03), 87.9% described it as unknown or slightly known; 42.4% launched it alone; 51.3% launched it with others.**
- For respondents reporting the proportion of code they contributed to their "current project," the upper-tail of the distribution is:

Proportions of code	***All 1055 Respondents***	***238 "Newbies"***
≥ 0.75	**44 %**	**54%**
≥ 0.95	**31 %**	**44%**

- Of 1451 respondents reporting code contributed to current projects, 58.9 % said ≥ 0.75 of their submitted code was included in the project's release version.

FLOSS-US developers in community-mode production have significantly different "motivation profiles" from those working on very small projects –

Identifier of five clusters	Participant's known project (s) are all 1-2 member **Independent**	> 29 member **Community**	Total
Cluster 1	11	8	19
%	5.14	4.55	4.87
Cluster 2	12	29	41
%	**5.61**	**16.48**	10.51
Cluster 3	102	79	181
%	47.66	44.89	46.41
Cluster 4	26	30	56
%	**12.15**	**17.05**	14.36
Cluster 5	63	30	93
%	**29.44**	**17.05**	23.85
Total	214	176	390
	100.00	100.00	100.00

Source: David, Shapiro and Waterman (2006)

Characterization of Cluster's Motivation Profile:

"Professionals": non-ideological, expert, self-employed or company-sponsored to collaborate on FLOSS projects

"Aspiring hackers": didn't need to modify existing code but like fixing bugs, learning about new pro- programs,

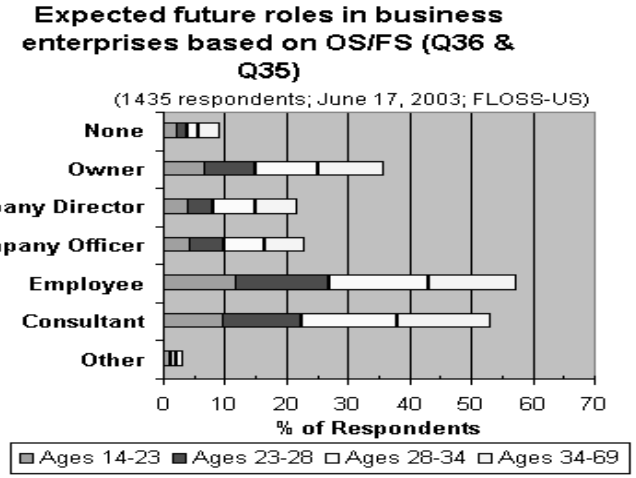
"Social learners": become better programmers, find out how programs work; work with like-minded, 'give back to the community', support *libre* ideology

"Raymondians": experienced hackers,,needed to modify existing code and fix bugs, use the code

"von Hippelites": modifying existing software not important, nor are learning or interacting with like-minded others; wanted to 'give back to the comm-unity' by launching their own project

Test of significant difference: *I-mode vs C-mode*: Pearson chi2(4) = 18.9175 Pr = 0.001

Nevertheless, the majority are looking forward to careers in FLOSS-based businesses….

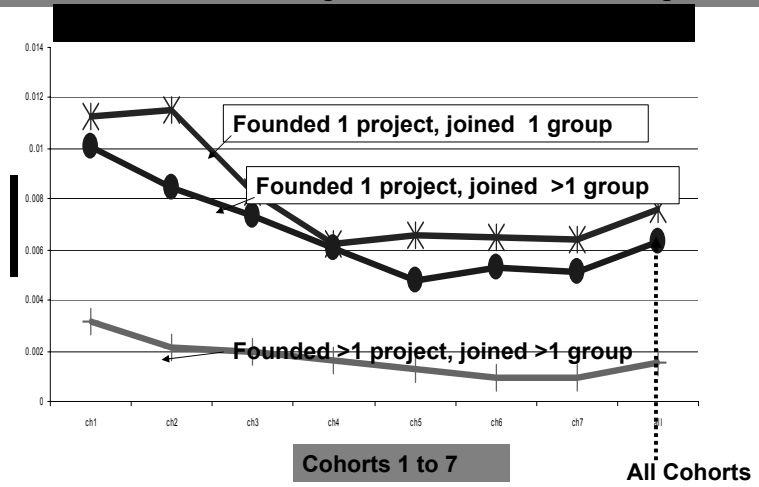


Source: *FLOSS-US Survey Report (2003)*

Sustaining the creation of new FLOSS projects

- Are platforms such as *SourceForge* functioning as virtual 'clusters' of innovative activity, stimulating new FLOSS projects, or do they grow mainly by attracting new start-ups?
- We have seen lots of projects appearing on *SourceForge* and other platforms, but launching new projects is harder than re-releasing pre-existing source code under an open source license. Do developers that register on these platforms continue to found new projects after they arrive?

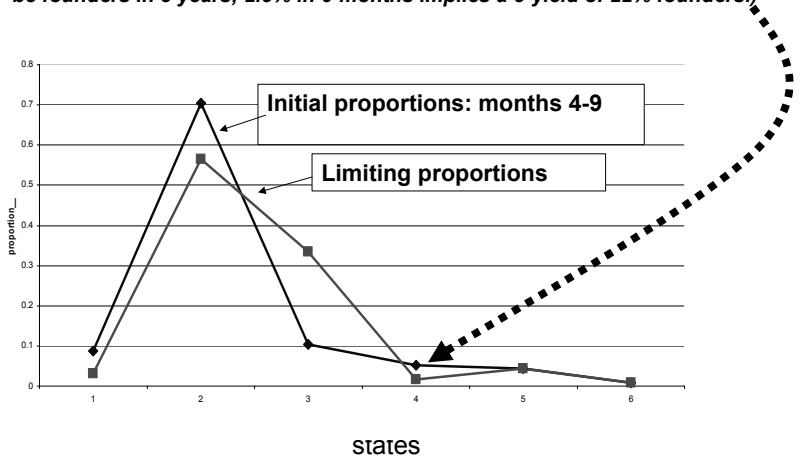
Proportions founding ≥1 projects in months 4-9 after registration on *SF.Net:*
Initial distributions of "active" registrants in successive arriving cohorts



The proportions of project founders declined at first but soon stabilized

Source: David and Pullani (2006)

Initial and "Limiting" Distributions of the Developers Registering on *SourceForge, Net* (during 1 Jan – 26 October 2001) Among 6 "Active" States ***(Note: 5% starting a project in each 6-month interval, implies 40% will be founders in 5 years; 2.5% in 6-months implies a 5-yield of 22% founders.)***



1= not member of a group (project); 2= non-founding member of group; 3=non-founding member of >1 group; 4=founder & member of 1 group; 5= founder of 1 & member of >1 group; 6 = founder & member of >1 group

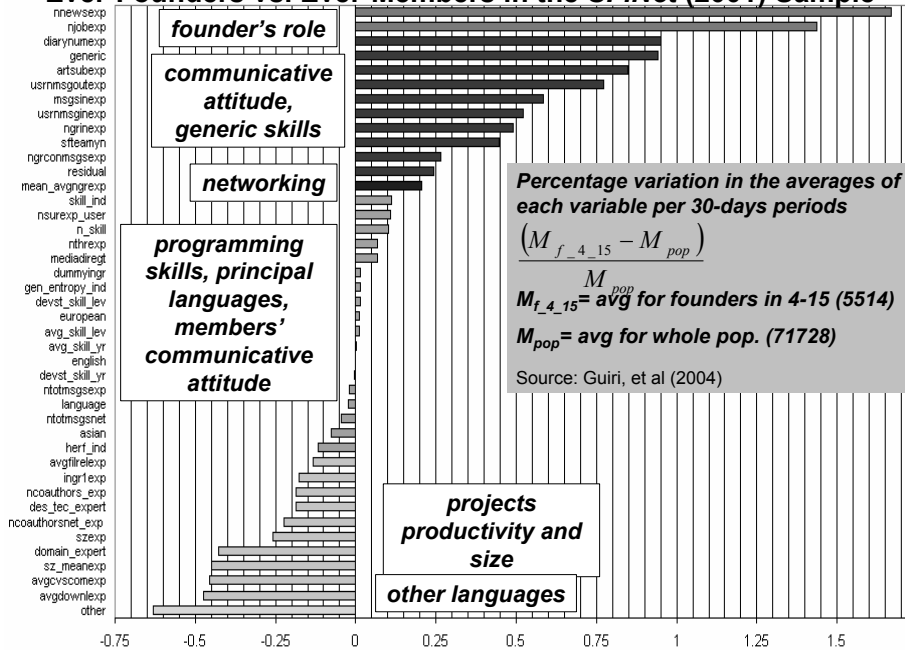
Project founding on *SourceForge* –

a small core among the "actives" become project "founders" within their first year, a majority of them within 3 years, if they haven't joined multiple group soon after arriving

Observed and expected proportions of *SF* population launching 1 or more projects within:	*all registrants in 2000-01:* avg. of **cohorts 1&2**	from **cohort 6**	*All "active" registrants in 2000-01:* avg. of **cohorts 1&2**	from **cohort 6**
months 4 - 9 post-registration	0.023	0.013	0.133	0.113
60 months post-registration	0.210	0.123	0.760	0.731
proportion of the registered *SF* sub-population that is *expected in the limit* to launch 1 or more projects within:	from the 10% sub-group of all registrants who were active during month 3 -- in "State 2"		from the 1% sub-group of all registrants who were active during month 3 -- in "State 3"	
a 6-month interval	0.030		0.000	
a 60- month interval	0.263		0.000	

Note: "State 2" = joining 1 project, launching 0 projects; "State 3" = joining >1 projects, launching 0 projects.
Source: David and Rullani (2006).

Ever-Founders vs. Ever-Members in the *SF.Net* (2001) Sample



Percentage variation in the averages of each variable per 30-days periods

$$\frac{\left(M_{f\_4\_15} - M_{pop}\right)}{M_{pop}}$$

$M_{f\_4\_15}$= avg for founders in 4-15 (5514)

M_{pop}= avg for whole pop. (71728)

Source: Guiri, et al (2004)

The evidence from *SourceForge* on the vitality of project founding is somewhat mixed—

A significant number of registrants among those who are at all active go on to found projects;

the observed rates of transitions from project membership to project launching have stabilized,

and the transition probabilities among the "states" imply "limiting" proportions involved in creating new projects that are about the same as those observed in months 4-9 following arrival on the platform.

But the project founders do not appear to have greater programming skills than the project members, and the are primarily involved in small and projects where there is less activity. In 2003 72% of the projects on SourceForge had only 1 member!

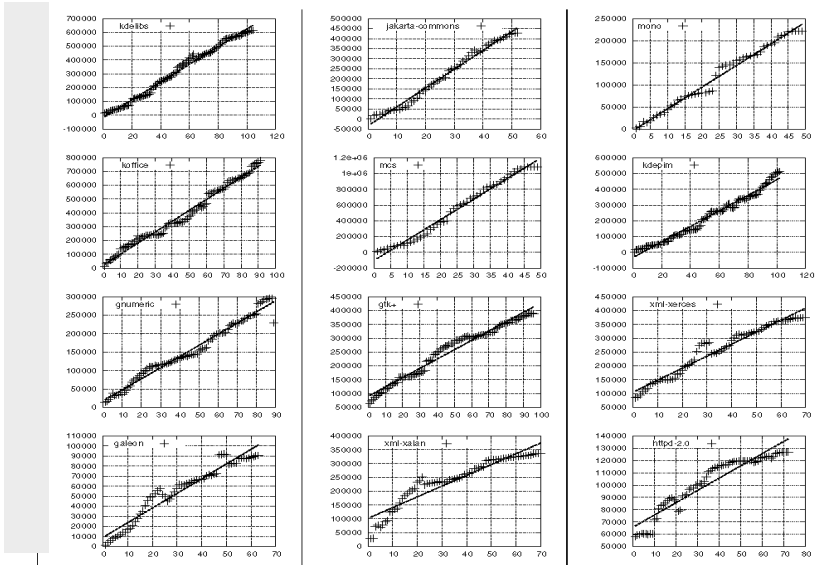
Sustaining maintainability as functionality (and size) grows in established projects

Sustaining maintainability and growth in large FLOSS projects

Won't the increase in the file size of these loosely managed projects result in the disproportionate increase of the interconnections among them?

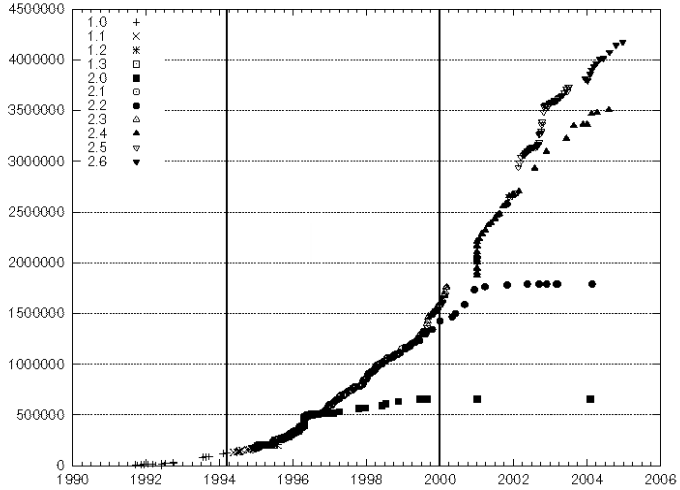
True: N files have N x (N-1) ≈ N^2 potential interconnections, and this could **create increasingly difficult problems for code maintenance and upgrading, causing greater and greater delays between releases of versions that significantly upgrade the functionally of the software system.**

But: that is hardly a necessary outcome.



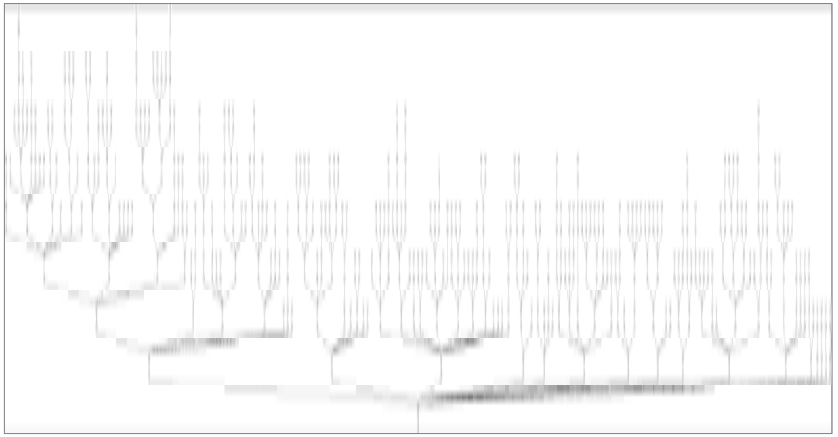
Linear growth of large projects' file sizes during months following version 1 release : 12 of 18 *libre source* projects Source: Robles et al (2005)

Super-linear growth of number of physical lines of code in the Linux kernel: versions 1.0 through 2.6



Source: Robles et al (2005): Figure 1. Growth (lines of code) of Linux

Semi-decomposable code architectures are "tree-like", facilitating flatter governance structures and reducing complexity of patching and upgrading



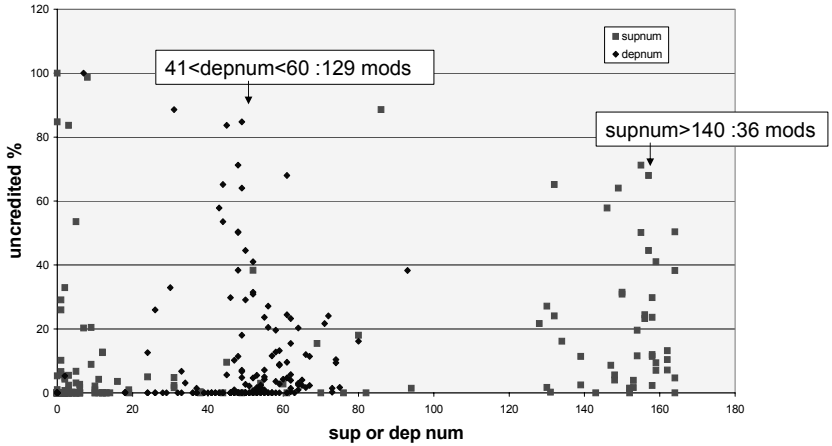
Source: David and Dalle (2005): Stochastic simulation of a highly evolved *code-tree*: 3 modules at *d* =14 from the root module

Dimension of the Linux kernel code-base

Linux kernel	Ver. 1.0	Ver.2.0.30	Ver.2.5.25
Approximate release date	Mar-94	Apr-97	Jul-02
Number of modules *	30	60	168
Number of files	593	2,155	12,451
Number of defined functions*	1,748	7,808	48,006
Physical lines of code*	121,987	527,773	3,157,543
Bytes of code (millions)	4.54	21.05	133.85
Number of *identified* authors*	158	616	2,263
% of code "un-credited" *	**18.8**	**12.2**	**14.9**

***See Ghosh and David (2003): modules or "code packages" defined for the LICKS study;"authors" identified by CODD algorithm from email signatures; "un-credited" bytes (KBOC)= CODD found no signature.**

linux 25 - uncredited % vs. dependency count measures



Supnum(i) = number of modules supported by ("call") the i-th module

Depnum(i) = number of modules that the i-th module depends on ("calls")

Source: David, Dalle, Ghosh and Wolak (2004)

A realistic optimist's conclusion

Correcting a number of frequently voiced grounds for pessimism doesn't answer all the sceptics,

because there are a number of serious challenges to the sustained success of the FLOSS community mode of production

– and that is the only viable alternative to conventional "closed" code production methods for complex software systems.

A realistic optimist's conclusion -- here are some of the serious challenges

- **Will the pool of programmers capable of writing good code be adequate? Are the many projects with only a few members indicative of difficulties in recruiting a talented core?**
- **Writing modular architectures are essential for the successful management and assimilation of new programmers who under the rationale of requirements and code design on big projects, but "code gods" who do that remain rare.**
- **The global FLOSS community is widely distributed and very loosely connected so that coordination on some strategies of resource allocation is difficult, and on this count is likely to be at a disadvantage if it continues to "engage" in head-to-head competition with focused commercial vendors who are learning "aigile programming" and moving toward continuous release production.**
- **The success of a number of large projects could make them targets for future patent infringement suits (like that of the SCO Group), pushing them toward protective limited liability incorporation as foundations, but the resulting controls are likely to conflict with the ethos of the original FLOSS movement.**
- **...for some others, see Brian Fitzgerald's "Has Open Source a Future?" in J. Feller et al., *Perspectives on Open Source Software* (2005)**

A realistic optimist's conclusion

Meeting these challenges will not automatically create a supportive commercial ecology for FLOSS projects, or guarantee steady funding from public and charitable sources.

But if addressing these issues is not sufficient for future success, it is likely to be necessary.

A Multi-dimensional View of "Sustainability" in Free & Open Source Software:

Commitment, Innovation, Maintenance and Growth

Paul A. David
Oxford Internet Institute & Stanford University
paul.david@oii.ox.ac.uk

A presentation to the OSS Watch Conference on
Open Source and Sustainability
held at the Said Business School, Oxford
10th -12th April 2006

INTRODUCTION

—An Economics Perspective on FLOSS

What is "open source" software?

- Free/Libre and Open Source Software (F/LOSS, or FLOSS) is distributed with the source code fully disclosed, and made available for use and modification without monetary charges.
- F/LOSS is software that is distributed under the terms of a special class of copyright licenses, primarily under the GNU General Public License (GNU GPL).

Reference: *The Open Source Definition* – version 1.9, available at: www.opensource.org/osd.html

(Originally written by Bruce Perren, this is now maintained by the Open Source Initiative.)

F/LOSS LICENSE CONDITIONS—IN BRIEF

"Free Software Licences" (FSF definition)

- **must include source code**
- **must allow distribution in source code as well as in compiled (machine code) form**
- **cannot restrict any party from distributing the software as a component of and aggregate distribution containing code from different sources**
- **must be "non-discriminatory" among licensees**
- **must ensure that programmers receive credit (blame) for their work**

"Copyleft" Software Licenses (FSF definition)

Authors who copy and distribute programs based on GPL'd code (derivative works) must do so under the GNU GPL license-- which contains the above provisions, plus this one.

What is so very interesting for an economist about the FLOSS development process ?

- Collective, distributed mode of creating (producing) an information-good: software
- Extensive voluntary participation by communities of skilled and neophyte software developers
- Novel use of IPR to distribute/publish software under "public domain-like" conditions
- Essential dependence of the production mode upon the "anti-proprietary" distribution regime
- Critical role of computer-mediated communications (CMC) for this production system
- Self-documenting nature of the process permits microlevel quantitative research on the organization of 'collective invention'

Four fundamental economic research questions about the FLOSS *Production Mode*

***At the Micro-level*:**

- **How are the human resource inputs mobilized?**
- **What kinds of inputs are supplied by participants in large project communities (*C-Mode* production – rather than *I-Mode*)?**
- **What factors motivate participants to devote effort to a particular project, and project task, rather than to other activities?**
- **How are these inputs allocated and coordinated within projects? (I.e., among tasks of a particular kind ,esp. coding, bug-fixing)**

Four fundamental economic research questions about the FLOSS *Production Mode*

***At the Meso-level*:**

- **How are resource inputs allocated among projects of various types?**
- **What mechanisms are used to govern projects, maintain code quality?**
- **How well does the system match software output to the needs of diverse users?**
- **What are the comparative costs in terms of resource use of FLOSS *C-Mode* production vs. closed-proprietary software?**

Starting from this perspective this presentation focuses on three (non-funding) dimensions of "sustainability"of FLOSS software development:

- **Sustaining commitment in FLOSS projects' development communities**
- **Sustaining rate of founding of new projects**
- **Sustaining maintainability as functionality (and size) grows in established projects**

Sustaining commitment in FLOSS projects' development communities

Is the initial enthusiasm and ideological commitment to "the movement" bound to dissipate?

Will those who were attracted to the movement begin seeking material rewards, and become disaffected by the few who are able "cash in" on their reputations as great project leaders?

Let's look at what the survey evidence can tell us about motivations among FLOSS developers.

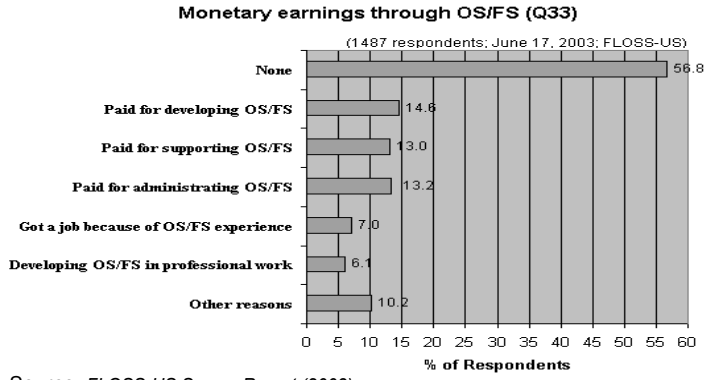
FLOSS-US
The Free/Libre/ Open Source Software Survey for 2003
To go immediately to the questionnaire, click here:

A Web Survey of Software Developers
conducted by the Stanford University (SIEPR) research project on

Economic Organization and Viability of Open Source Software
With funding support from the National Science Foundation.
http://siepr.stanford.edu/programs/OpenSoftware_David/OS_Project_Funded_Announcmt.htm

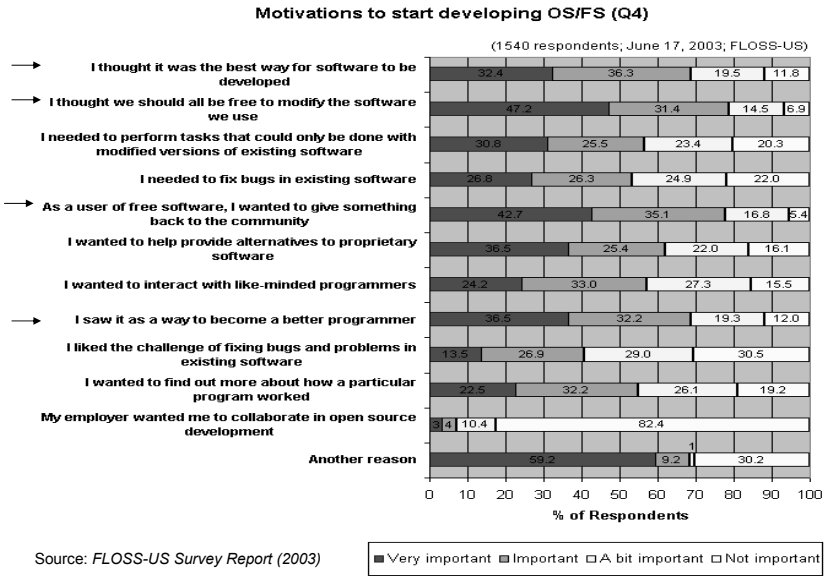
The FLOSS-US Survey: First Report
(September 2003) is available at:
www.stanford.edu/group/floss-us/report/FLOSS-US-Report.pdf

Income earnings from FLOSS activities remains atypical among survey respondents: 57% of FLOSS-US respondents cite not having direct or derived earnings benefits from their activities.



Source: *FLOSS-US Survey Report (2003)*

Ideological and self-improvement motives are salient among initial motivations of FLOSS-US (2003) developers



Source: *FLOSS-US Survey Report (2003)*

Motivations are not static -- they undergo change, and the question is whether they change in ways that are likely to allow them to be fulfilled by the experience of participating in FLOSS activities:

Initial Motivational Groups	n	%
Enthusiasts	**104**	**3.7**
Software improvers	**285**	**10.2**
Recognition seekers	**308**	**11.1**
Materialists	**327**	**11.7**
Ideologists	**472**	**17.0**
'Triers' (= diffuse motives)	**1288**	**46.3**
Total	**2784**	**100.0**

© 2004 International Institute of Infonomics / Merit
Source: R. Glott et al., Motivations of Free/Libre and Open Source Developers. May 2004.

This is the distribution of FLOSS-EU Survey (2002) respondents among the main motivational groupings identified by principal components analysis of their reasons for initially participating in open source software development….

…and this is they way motivation profiles of the same developers appear to evolve: stated reasons for continuing to work on FLOSS projects do not shift towards pursuit of economic rewards

Flows from initial to continuing motivational groups within the FLOSS community

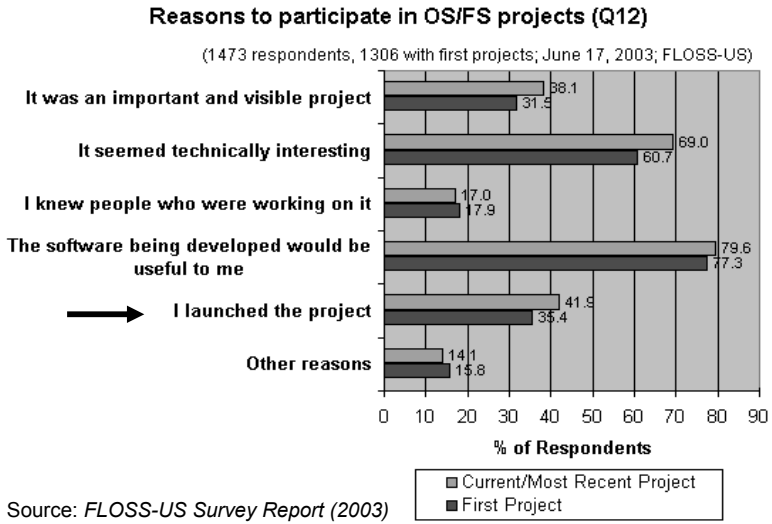
		INITIAL MOTIVATIONAL GROUPS						
		enthusiasts	software improvers	recognition seekers	materialists	ideologists	'triers'	total
CONTINUING MOTIVATIONAL GROUPS	recognition seekers	33.7	6.0	41.6	14.1	7.2	6.8	12.0
	skills improvers	32.7	22.8	37.0	26.3	43.6	31.1	32.5
	software improvers	30.8	37.5	13.0	39.4	14.4	23.2	24.2
	ideologists	2.9	33.7	6.4	20.2	34.7	39.9	31.2
	total	100	100	100	100	100	100	100

Figures are percentages of the initial motivational groups. © 2004 International Institute of Infonomics / Merit

Source: R. Glott, R. A. Ghosh, B.L. Krieger, Motivations of Free/Libere and Open Source Software Developers, May 2004

The big group that cited diffuse motives ('triers') for beginning FLOSS have redistributed themselves toward "skill-improvement" and "ideological" reasons for continuing – motivation clusters that represent 64% of total respondents.

FLOSS-US respondents explained their project *choices* in terms more pragmatic than the reasons they gave for participating in FLOSS development activities



Source: *FLOSS-US Survey Report (2003)*

The importance of "the personal utility of the software" among the reasons given by FLOSS-US (2003) respondents for their current project choices reflects the predominance of very small, *I-mode* projects that the respondents had personally launched.

- Of 1473 respondents listing a "current project", 64.8% described it as "unknown" or "slightly known': 33.0% launched it alone;
  : 46.8% launched it with others.
- Of 1306 respondents listing their "first projects", 61.7% described it as "unknown" or "slightly known"; 35.4% launched it with others.
- **Of 238 "newbies" (those starting a "first &current" project in 2001-03), 87.9% described it as unknown or slightly known; 42.4% launched it alone; 51.3% launched it with others.**
- For respondents reporting the proportion of code they contributed to their "current project," the upper-tail of the distribution is:

Proportions of code	***All 1055 Respondents***	***238 "Newbies"***
≥ 0.75	**44 %**	**54%**
≥ 0.95	**31 %**	**44%**

- Of 1451 respondents reporting code contributed to current projects, 58.9 % said ≥ 0.75 of their submitted code was included in the project's release version.

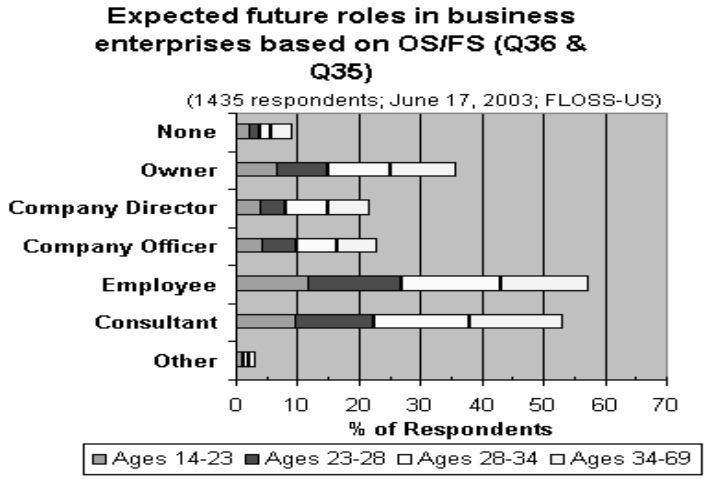
Distributions of motives of participants in *C-mode* and I-mode projects differ

Distribution of Developers by "Motivation Profiles" (from Cluster Analysis of Sub-samples the FLOSS-US Survey Respondents – Grouped by Projects' Membership Sizes (>30) vs. (1-2))

Clusters	I-Mode	C-Mode	Total	Characterization of cluster's motivation profile
Cluster 1 *%*	11 *5.14*	8 *4.55*	19 *4.87*	Non- ideological, expert, self- employed or company-sponsored to collaborate on FLOSS projects: **professionals.**
Cluster 2 *%*	12 ***5.61***	29 ***16.48***	41 *10.51*	*Didn't* need to modify existing code but like fixing bugs, learning new programs: **aspiring hackers.**
Cluster 3 *%*	102 *47.66*	79 *44.89*	181 *46.41*	Become better programmers, find out how programs work; work with like-minded, 'give back to community', support Libre ideology: **social learners.**
Cluster 4 *%*	26 ***12.15***	30 ***17.05***	56 *14.36*	Needed to modify existing code and fix bugs, see open source as the "best way": **experienced hackers.**
Cluster 5 *%*	63 ***29.44***	30 ***17.05***	93 *23.85*	Modifying existing software is *not* important, *nor* are learning, and interacting with like-minded others, wanted to 'give back to community', many launched their own projects: **(individualistic) user-innovators.**
Total *%*	214 *100*	176 *100*	390 *100*	

Source: David, Shapiro and Waterman (2006)

Nevertheless, the majority are looking forward to careers in FLOSS-based businesses….

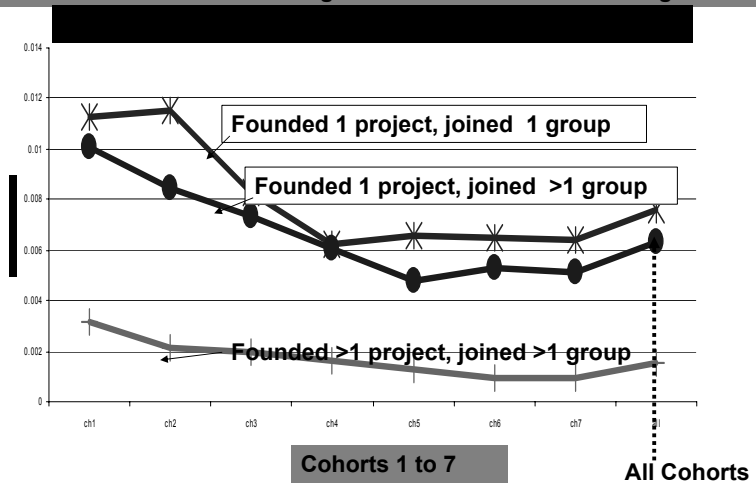


Source: *FLOSS-US Survey Report (2003)*

Sustaining the creation of new FLOSS projects

- Are platforms such as *SourceForge* functioning as virtual 'clusters' of innovative activity, stimulating new FLOSS projects, or do they grow mainly by attracting new start-ups?
- We have seen lots of projects appearing on *SourceForge* and other platforms, but launching new projects is harder than re-releasing pre-existing source code under an open source license. Do developers that register on these platforms continue to found new projects after they arrive?

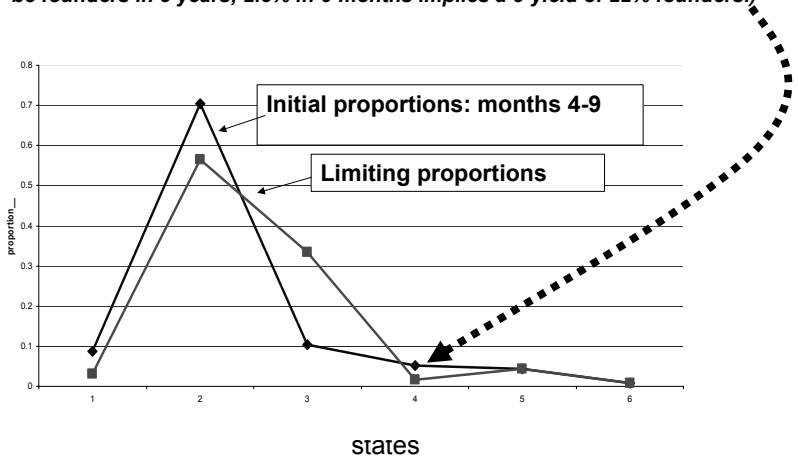
Proportions founding ≥1 projects in months 4-9 after registration on *SF.Net:*
Initial distributions of "active" registrants in successive arriving cohorts



The proportions of project founders declined at first but soon stabilized

Source: David and Rullani (2006)

Initial and "Limiting" Distributions of the Developers Registering on *SourceForge, Net* (during 1 Jan – 26 October 2001) Among 6 "Active" States *(Note: 5% starting a project in each 6-month interval, implies 40% will be founders in 5 years; 2.5% in 6-months implies a 5-yield of 22% founders.)*



1= not member of a group (project); 2= non-founding member of group; 3=non-founding member of >1 group; 4=founder & member of 1 group; 5= founder of 1 & member of >1 group; 6 = founder & member of >1 group

Source: David and Rullani (2006)

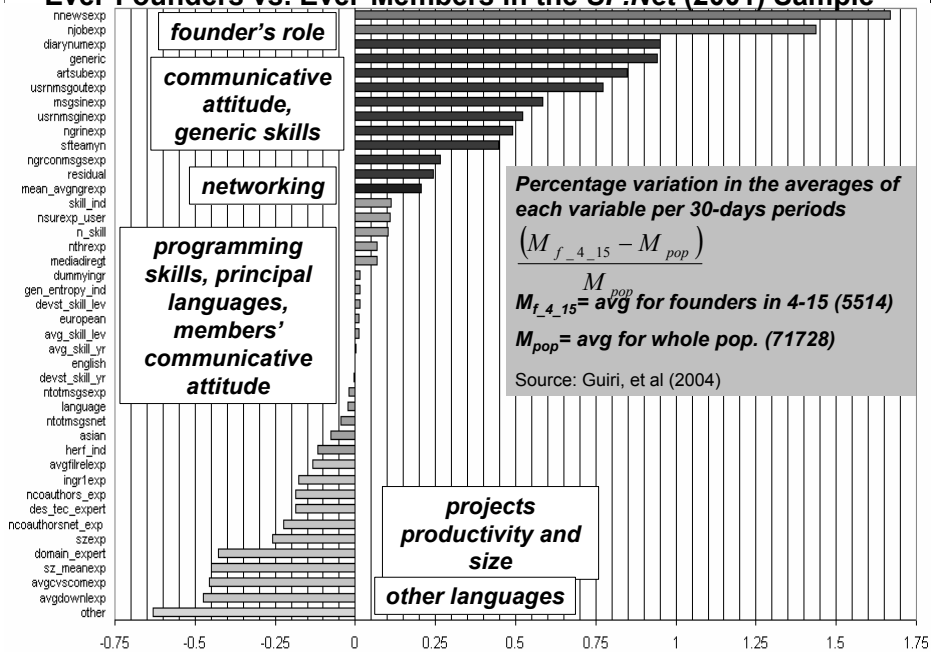
Project founding on *SourceForge* –
a small core among the "actives" become project "founders" within their first year, a majority of them within 3 years, if they haven't joined multiple group soon after arriving

Observed and expected proportions of *SF* population launching 1 or more projects within:	*all registrants in 2000-01:* avg. of **cohorts 1&2**	from **cohort 6**	*All "active" registrants in 2000-01:* avg. of **cohorts 1&2**	from **cohort 6**
months 4 - 9 post-registration	0.023	0.013	0.133	0.113
60 months post-registration	0.210	0.123	0.760	0.731

proportion of the registered *SF* sub-population that is *expected in the limit* to launch 1 or more projects within:	from the 10% sub-group of all registrants who were active during month 3 -- in "State 2"	from the 1% sub-group of all registrants who were active during month 3 -- in "State 3"
a 6-month interval	0.030	0.000
a 60- month interval	0.263	0.000

Note: "State 2" = joining 1 project, launching 0 projects; "State 3" = joining >1 projects, launching 0 projects.
Source: David and Rullani (2006).

Ever-Founders vs. Ever-Members in the *SF.Net* (2001) Sample



The evidence from *SourceForge* on the vitality of project founding is somewhat mixed—

A significant number of registrants among those who are at all active go on to found projects;

the observed rates of transitions from project membership to project launching have stabilized,

and the transition probabilities among the "states" imply "limiting" proportions involved in creating new projects that are about the same as those observed in months 4-9 following arrival on the platform.

But the project founders do not appear to have greater programming skills than the project members, and the are primarily involved in small and projects where there is less activity. In 2003 72% of the projects on SourceForge had only 1 member!

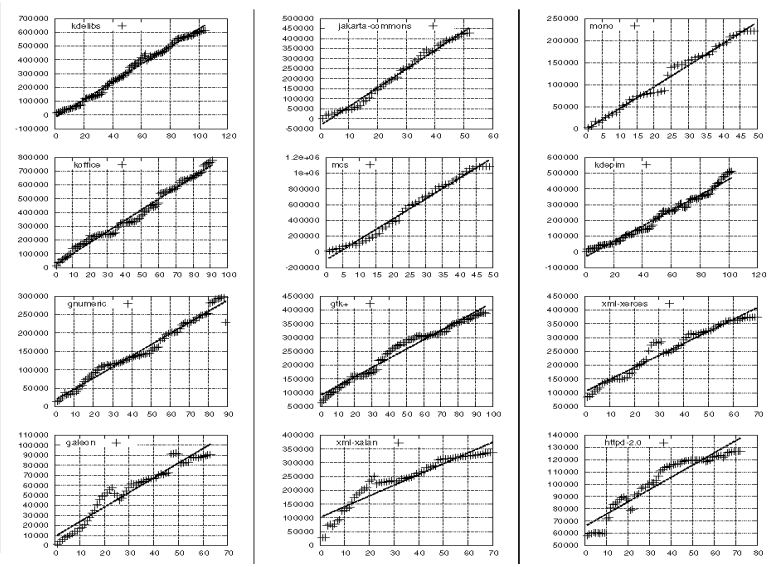
Sustaining maintainability as functionality (and size) grows in established projects

Sustaining maintainability and growth in large FLOSS projects

Won't the increase in the file size of these loosely managed projects result in the disproportionate increase of the interconnections among them?

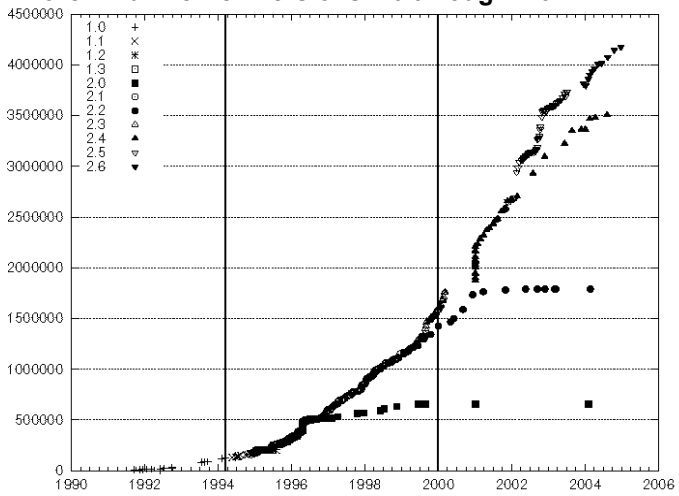
True: N files have N x (N-1) ≈ N^2 potential interconnections, and this could **create increasingly difficult problems for code maintenance and upgrading, causing greater and greater delays between releases of versions that significantly upgrade the functionally of the software system.**

But: that is hardly a necessary outcome.



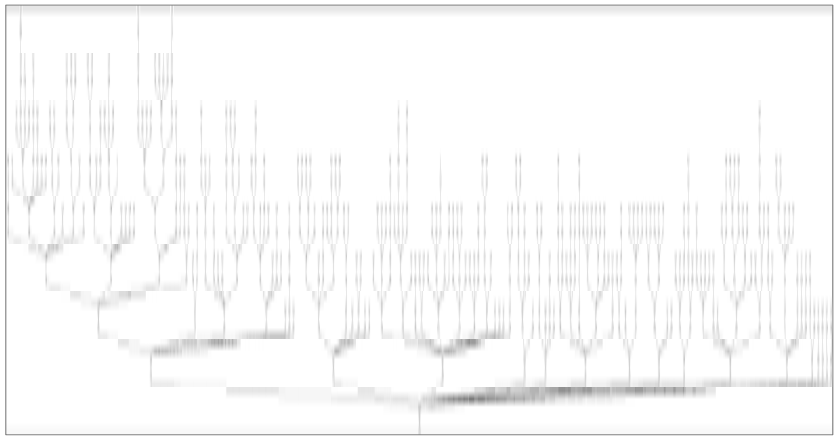
Linear growth of large projects' file sizes during months following version 1 release : 12 of 18 *libre source* projects Source: Robles et al (2005)

Super-linear growth of number of physical lines of code in the Linux kernel: versions 1.0 through 2.6



Source: Robles et al (2005): Figure 1. Growth (lines of code) of Linux

Semi-decomposable code architectures are "tree-like", facilitating flatter governance structures and reducing complexity of patching and upgrading



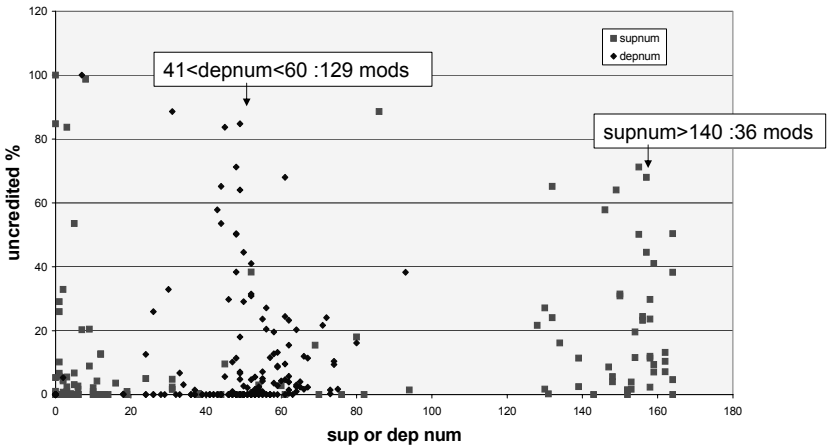
Source: David and Dalle (2005): Stochastic simulation of a highly evolved *code-tree*: 3 modules at *d* =14 from the root module

Dimension of the Linux kernel code-base

Linux kernel	Ver. 1.0	Ver.2.0.30	Ver.2.5.25
Approximate release date	Mar-94	Apr-97	Jul-02
Number of modules *	30	60	168
Number of files	593	2,155	12,451
Number of defined functions*	1,748	7,808	48,006
Physical lines of code*	121,987	527,773	3,157,543
Bytes of code (millions)	4.54	21.05	133.85
Number of *identified* authors*	158	616	2,263
% of code "un-credited" *	**18.8**	**12.2**	**14.9**

***See Ghosh and David (2003): modules or "code packages" defined for the LICKS study;"authors" identified by CODD algorithm from email signatures; "un-credited" bytes (KBOC)= CODD found no signature.**

linux 25 - uncredited % vs. dependency count measures



Supnum(i) = number of modules supported by ("call") the i-th module

Depnum(i) = number of modules that the i-th module depends on ("calls")

Source: David, Dalle, Ghosh and Wolak (2004)

A realistic optimist's conclusion

Correcting a number of frequently voiced grounds for pessimism doesn't answer all the sceptics,

because there are a number of serious challenges to the sustained success of the FLOSS community mode of production

– and that is the only viable alternative to conventional "closed" code production methods for complex software systems.

A realistic optimist's conclusion -- here are some of the serious challenges

- **Will the pool of programmers capable of writing good code be adequate? Are the many projects with only a few members indicative of difficulties in recruiting a talented core?**
- **Writing modular architectures are essential for the successful management and assimilation of new programmers who under the rationale of requirements and code design on big projects, but "code gods" who do that remain rare.**
- **The global FLOSS community is widely distributed and very loosely connected so that coordination on some strategies of resource allocation is difficult, and on this count is likely to be at a disadvantage if it continues to "engage" in head-to-head competition with focused commercial vendors who are learning "aigile programming" and moving toward continuous release production.**
- **The success of a number of large projects could make them targets for future patent infringement suits (like that of the SCO Group), pushing them toward protective limited liability incorporation as foundations, but the resulting controls are likely to conflict with the ethos of the original FLOSS movement.**
- **…for some others, see Brian Fitzgerald's "Has Open Source a Future?" in J. Feller et al., *Perspectives on Open Source Software* (2005)**

A realistic optimist's conclusion

Meeting these challenges will not automatically create a supportive commercial ecology for FLOSS projects, or guarantee steady funding from public and charitable sources.

But if addressing these issues is not sufficient for future success, it is likely to be necessary.