2 Understanding Free Software Developers: Findings from the FLOSS Study

Rishab Aiyer Ghosh

This chapter presents an overview of findings from the Survey of Developers from the FLOSS (Free/Libre/Open Source Software) project, involving over 2,700 respondents from among free/open source software developers worldwide. The survey studied several factors influencing developers' participation within the free/open source community, including their perceptions of differences within the community and with the commercial software world; personal, ethical, political, and economic motives for participation; and their degree of interaction within and contribution to the free/open source software community. These results are linked to preliminary findings from a study of developer contribution to the Linux kernel based on an analysis of the source code.

The Need for Empirical Data

The phenomenon of Free/Libre/Open Source Software[1]—the development of software through collaborative, informal networks of professional or amateur programmers, and the networked distribution making it available to developers and end-users free of charge—has been widely and well documented. Editorials and policy papers have been written on the impact of the free software movement on the computer industry, business in general and the economy at large. However, few models have been developed that successfully describe, with supporting data, why or how the system works, or that explain the functioning of collaborative, productive networks without primary dependence on money.

The common speculation regarding the monetary worth of such collaborative development has translated into widely fluctuating share prices for the companies that have devoted much of their business plans to the free software philosophy. But hard data on the monetary value generated by this phenomenon is almost nonexistent. Indeed, hard data on *any* aspect

of the FLOSS phenomenon is rare. One reason for this lack of empirical data might be that in a very short period of time, this phenomenon has attracted a lot of research attention. Researchers and practitioners have tended to be quick to state that open source is (or is not) a revolutionary "new" form of something—programming, economic production, or social interaction. These explanations have generally been based on anecdotal evidence or very small sample data, which doesn't make them wrong—just hypothetical.[2]

Given that most models and techniques for economic evaluation and measurement require the use of money, nonmonetary economic activity, such as the creation and distribution free software, is left unmeasured, at least in any usefully quantifiable sense. Although there are studies and models for quantitative analysis of nonpriced goods (e.g., the measurement of knowledge), in an economy they tend to be useful primarily in judging the influence of such goods within organizations, markets, or other socioeconomic structures for which the common forms of measurement are clearly dominated by monetary indicators. Measurement is far more complex and ambiguous in a context where the essential and primary economic activity—the generation of free software through collaborative networks—is unusual in its avoidance of the use of money as mode of exchange.

The lack of empirical data is not surprising, though—it is extremely hard to collect, for several reasons. First, without monetary measures, other indicators of developers' activity have to be used (indeed, defined in order to be used). While there may be some quantitative indicators that are objective in nature,[3] the lack of objective "census-type" sources means that many indicators, quantitative or qualitative, may require the use of surveys.

Who Can We Survey?

This leads to an immediate problem: there is no universal, clearly recognized, objective data on the population to be surveyed. While seemingly obvious, it bears emphasizing that there is no clear definition of a free software developer (other than "someone who writes free software code"); there is no universal list of all developers; there is no accurate information on the number of developers in existence or the growth in this number.[4] There is no census or national accounts database that lists the distribution of developers by country of residence, age, income level, or language.

The lack of a basic data set on the universal population, something that is taken for granted in surveys of many other groups, is unavailable for

FLOSS software developers, with the result that attempts to fill in gaps in empirical factual data through surveys require a choice from among three types of surveys:

1. Survey responses that might be indicative of the general population, but provide reliable data only on the actual respondents.
2. Survey responses that reflect a predefined subset of the general population, providing insights into that subset, but certain responses might be a consequence of the predefinition (preselection) process and thus not reflect the general population.
3. Survey respondents are drawn randomly from the general population; thus, while responses reflect the general population as a whole, they might not be representative of the general population for certain (especially demographic) criteria.

For completeness, the fourth option, which is ideal but unavailable, is also listed:

4. Survey respondents are drawn from the general population in order to be representative of the general population for certain criteria; for example, age or nationality, thus leading to responses that reflect the general population and also follow the distribution (based on the representation criteria) of the general population.

The BCG/OSDN survey (Boston Consulting Group 2002) is an example of the second approach. By prescreening respondents and inviting their response by e-mail, it clearly defined the subset of the general population from which the sample was drawn. As such, responses can be representative of the pre-screened subpopulation, and even weighted in order to truly reflect the defined population subset. However, the results say little about the general population beyond the defined subset, as it was not sampled at all. Indeed, some results, such as nationality, years of experience, or political attitudes might result from the preselection criteria,[5] and though these results provide interesting detail on the sub-population, they cannot be generalized to apply to the universal free software developer.

For the FLOSS developer survey, we chose the third option.

Secondary Sources

Empirical data on FLOSS developers is not only difficult to collect, but once collected might also be somewhat unreliable. This is perhaps a reason to try to find secondary sources to match subjective empirical data, and methods of validating them—it also provides a handy excuse for papers that do not cite empirical data!

Such secondary sources include not just objective data resulting from analysis of source code,[6] but also more conventionally reliable surveys of, for instance, institutions that use free software. Such surveys can be conducted in an orthodox fashion, using industrial databases or public census records to build stratified representative samples, as was done in the FLOSS Survey of User Organisations.[7] Institutional surveys are immensely useful in themselves, to study the large-scale use of free software, but they can also provide data on organizations' relationships with developers that, with some "triangulation," can corroborate the results of developer surveys.

Models and Hypotheses

The FLOSS developer survey aimed to gather data that would support (or refute) the several anecdote-based models of FLOSS development that exist. Broadly, hypothetical models of free software developers attempt to interpret the motives of developers, determine the structure of their interaction, and predict their resulting individual and collective behaviour. (Some models are less concerned about precise motives; Ghosh (1998a) only claims that altruism is not a significant motive, and Benkler (2002) argues that specific modes of organization, rather than motives, are important.)

Our survey was designed to test the assumptions made by many models and to collect a set of data points that could be used to validate or improve such models.

Assumed Motivations

Are developers rational, and if so, are they altruistic, or self-interested? Are they driven by a profit motive, and if so, is it monetary or non-monetary? Do they want to become famous? Are they writing free software to signal their programming proficiency in the job market? Or do they just want to fix a problem for their own use? Are they mainly interested in having fun? Is programming artistic self-expression? Or is it self-development, a distributed university? Or is it a community experience, where the pleasure is in giving to others? Are developers politically driven, wanting to destroy large software companies or the notion of proprietary software and intellectual "property"?

Most models of FLOSS development assume one or another of these motives as the key driver. In fact, it turns out, the truth is all of the above, combined in different proportions for different people. The survey ques-

tionnaire had several questions dealing with motivation, some of them with overlapping responses; people don't always think consciously about their motives, and repetition and different phrasing help draw out more data and add perspective.

Assumed Organization

When any group of individuals interacts, the structure of their interaction is a strong determinant of their effectiveness and output. Organizational structure may be a result of the motives of individual participants; it may also prompt or create certain motives. For example, in a strictly hierarchical organization, getting to the top may be an important motive; in a flat organization, being at the top may have fewer benefits and thus be less motivating.

Organizational structure is a predictor of behavior, though, and many models of free software development use an assumed organizational structure in order to predict behavior. Benkler's (2002) "peer production" depends fundamentally on structural assumptions; Lerner and Tirole's "Simple Economics" (chap. 3, this volume) does not directly depend on a certain organizational structure in FLOSS developer communities, but requires that the structure facilitate the spread of reputation (as signaling) within the community and to the employment market (thus also assuming a meta-structure linking the FLOSS community to the priced economy directly via the job market). The "cooking-pot" model and the "bazaar" (Raymond 2001), though, are more tolerant of different organizational structures.

In the context of free software developer communities, one could classify organizational structure on the axes of hierarchy, modularity, and connectivity (see table 2.1). Here, *modular/integrated* refers to the extremes in the integrated nature of the production, while *connectivity* refers to the integrated nature of the interaction (or social links, which might or might not lead to integrated products), so this is not just the organizational structure of a community, but of a *productive* community. Of course the examples given in each box are somewhat arbitrary, as not all boxes can be reasonably filled in and retain relevance to the context of FLOSS development.

Organizational structure can be determined to some extent (and even objectively) by analyzing source code, which allows the measurement of modularity, but also the degree of connectivity and even hierarchy (using concentration of contribution as a proxy) through the identification of authors and clusters of authorship.[8]

Table 2.1
A classification of organizational structures

	Modular, connected	Integrated, connected	Modular, nonconnected	Integrated, nonconnected
Hierarchy	Cabal/inner circle (bazaar), "Simple Economics" (signaling)	Commercial (cathedral)	Benevolent Dictator (bazaar)	Commercial
Flat	"Peer production" with reputation, "Cooking-pot market"	"Hive mind"	"Peer production" (bazaar)	?

Otherwise, determining organizational structure empirically on a large scale is hard.[9] Through a developer survey, it can be estimated by asking about the number of projects a developer participates in, degree of collaboration with other developers, and leadership positions.

Assumed Behavior

Behavior, individual or collected, is what models aim to predict, so empirical data plays a valuable role here in testing the validity of models. Most such data is likely to result from objective studies of developer communities, such as the dynamic study of the developers and components of the Linux kernel in the LICKS project.[10] From developer surveys, especially if conducted repeatedly over time, it would be possible to determine whether certain types of behaviour occur as predicted by models, or indeed as predicted by developers' own responses to other questions.

This is useful in order to validate the strength of reported motivations; for example, it is harder to believe those who claim money is not a motivation if they also report high earnings from their FLOSS software. At the very least, one might expect (from the point of view of consistency) that developer's motives change over time based on the rewards (or lack thereof) they receive through their efforts. Indeed the FLOSS developer survey attempts to measure this dynamic aspect of motivation by asking what motivated developers when they first joined the community, in addition to their motivations for continuing participation. As both questions are asked at the same time, this response relies on developers' memory.

Other measurable aspects of behavior relate to developers' planned future actions (as reported); for instance, with relation to the job market. Later sections elaborate on this topic through the lens of the FLOSS survey.

Conducting the FLOSS Developer Survey

Methodology and Sampling

Early in the design of the FLOSS survey methodology, we faced the question of whether it is possible to survey a representative sample of developers. The question actually has two parts: is it possible to ensure that respondents are developers, and is it possible to identify a sample that is representative of developers based on some filtering criteria? We'll address take the second question first.

Our conclusion, as described previously, was that there is insufficient empirical data on FLOSS software developers to identify the criteria of sampling. However, without empirical data as a basis, it is not possible to demonstrate that a chosen sample of respondents is representative of developers in general: that is, it is impossible to sample developers and know with any confidence that the distribution of nationalities, age, or income levels is representative of the distribution in the total (unsampled) population of developers.

Therefore, we decided that in order to have results with empirical validity for the universal population of developers; we would have to attempt a random sample. The survey was self-distributing that is, it was posted to various developer forums, and then reposted by developers to other forums, many of which are listed in the FLOSS survey report's (http://flossproject.org/) Web site. The survey announcement was translated into various languages in order to correct possible biases inherent in an English-language survey announced only on English-language websites.[11] We can state confidently that the survey was seen by a very large percentage of all developers (it was announced on Slashdot, among other places) and therefore the sample that chose to respond was random, though with some identifiable bias (including, as with any voluntary survey, self-selection).

Having drawn a random sample, we had to ensure that we were indeed drawing a sample of actual developers. We were able to do so through the validation process described in the following section.

Response Rate and Validation

One of the requirements of open, online questionnaires is verifying that the respondents really belong to the group that is under scrutiny. The

survey definition of the universal developer population is "everyone who has contributed source code to a free/libre/open source software package." Our respondents belong to this population as follows:[12]

- We asked respondents to provide complete or partial e-mail addresses for validation purposes.
- We matched these e-mail addresses to names or e-mail addresses found in the source code analysis (see Ghosh et al. 2002, part V) or matching them to sources on Internet archives.
- This subsample of 487 respondents individually identified as certain developers were compared to the rest of the respondents by statistically comparing their responses. This process involved a comparison of means and standard deviations of the two groups (known developers and other respondents) with regard to a selection of variables of our data set. The result showed very little statistical difference. In the very few responses where minor differences existed, we found that the group of verified FLOSS developers consisted of slightly more active and professionally experienced persons.

The FLOSS developer survey received 2,774 responses. Due to the page-wise design of the questionnaire, where responses to the first page were recorded even as the next page of questions were presented, this figure represents the respondents who answered the first set of questions, while there were 2,280 responses to the entire questionnaire.

What Do We Know Now?

Demographics: Married with Children?

A presentation of demographics is usually the starting point for an analysis of any survey. For the reasons given earlier, the FLOSS methodology does not provide a sampling that is representative of developer demographics, especially of the geographic distribution of developers. (A comparison of geographical data from different developer surveys, as well as much else that this chapter draws on, is in Ghosh et al. 2003.) It is possible to discuss other features of developer demographics, though, for which the FLOSS survey is likely to be more representative, such as age, gender, and civil status (which we presume have a similar distribution across different nationalities).

Almost all (98.8 percent) of respondents to the FLOSS developer survey are male. This is similar to the 98.6 percent reported as male in the WIDI survey (from a much larger sample size, of nearly 6,000; see Robles-

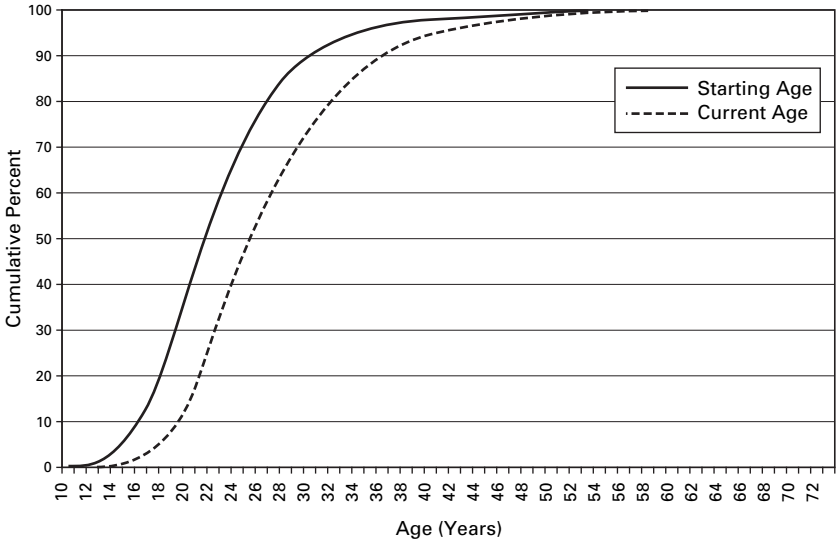

Figure 2.1
Current and starting age of developers (© 2002 International Institute of Infonomics, FLOSS developer survey)

Martínez et al. 2001) and 98 percent reported as male in the BCG survey (chap. 1, this volume). It should be noted that the surveys probably underrepresent female developers. As self-selecting surveys, they are dependent on developers who chose to respond to the survey itself, and to the specific question on gender. However, given the degree of coincidence on this point across three quite different surveys, it would seem unlikely that female participation in the FLOSS developer community is much higher than 5–7 percent.

The FLOSS survey showed developers are quite young, with more than 60 percent between the ages of 16 and 25. Figure 2.1 shows the cumulative percentage for developers' age when they first started free software development, compared with developers' current age. Because we asked when respondents first started development and their age at that time, we were able to calculate two age points for each developer as well as identify the peak year for the start of development—2000 (the survey was carried out in early 2002).

Despite the apparent youth of developers, as figure 2.2 shows, single developers are in a minority (albeit a large one, 41.4 percent) and about the same as the surprisingly large fraction (39.9 percent) who live together

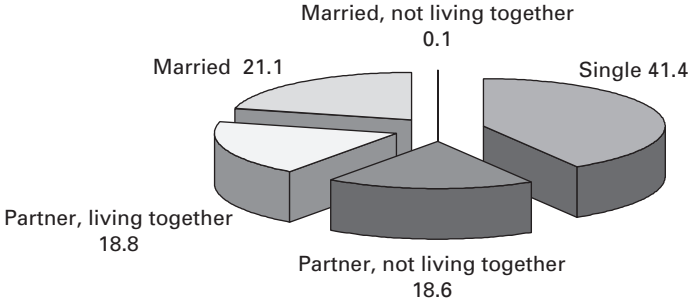

Figure 2.2
Civil status of developers

with a partner or spouse. Again, going against the nerdish stereotype, 17 percent of developers reported having children. Half of those have one child. There is a correlation between having children (or even being married) and having a stronger career-oriented participation in development. Most children were reported to be under two years of age at the time of the survey. One might thus wonder whether as the free software baby boom coincided with the dot-com peak, developers saw their earning possibilities soar and took on financial and familial responsibilities.

Motivation

The FLOSS survey addressed the complex issue of what motivates developers to contribute to the community in a series of multidimensional questions. This aspect in particular is elaborated in much detail in Ghosh et al. 2003, so only brief summary is presented here.

When I started writing about free software—and "non-monetary economic"[13]—phenomena in the mid-1990s, there was widespread suspicion among traditional economists and others that this was a domain either for hobbyists or for irrational groups of people mainly driven by warm fuzzy feelings of communal sharing and gifts. Since the idea of a gift is usually associated with knowing the recipient, and the politics of developers in particular tend towards the libertarian rather than the communitarian, the notion of "gift economy"[14] might seem unjustified. As open source became a hot topic for investigation in the social sciences, recent hypotheses usually supposed largely rational, self-interested motives, among the most extreme being that open source is explained by the "simple economics" of signaling for better career prospects and hence monetary returns.[15]

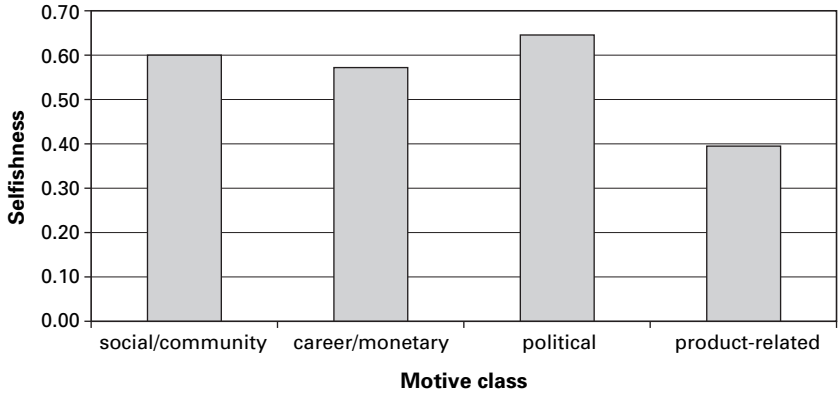

Figure 2.3
"Selfishness" or "profit-seeking measure" of developers

In the absence of clear monetary transactions, the interplay of contribution and return can be described in the form of "balanced value flow"[16] where one assumes rational self-interest but allows that self-interest can include a range of different types of reward, not just monetary compensation. While the FLOSS survey attempts to measure some of these possible rewards, a simple question is to ask whether individual developers value their own contribution more or less than their perceived rewards; that is, "I give more than/less than/the same as I take" in relation to the developer community at large.

We asked exactly this, and the resulting "selfishness measure" or "altruism measure" is shown in figure 2.3, for respondents falling into four motivation categories (as described later in this section). This measure ranges from –1 (purely altruistic) and +1 (purely selfish) and is calculated as the difference between the "selfish" responses ("I take more than give ") and the "altruistic" responses ("I give more than I take"), as a fraction of total responses.[17]

What we see is that more respondents are selfish than altruistic from all motive classes. Indeed, 55.7 percent of all developers classified their relationship with the community as "I take more than I give" and a further 14.6 percent felt their contribution and reward was balanced; only 9 percent could be classed as consciously altruistic in that they reported that they give more than take. It should be noted that this measure does not reflect selfishness or altruism as *intent*—which is what would be correct—but as an *outcome*, in which case the words don't fit as accurately. Thus,

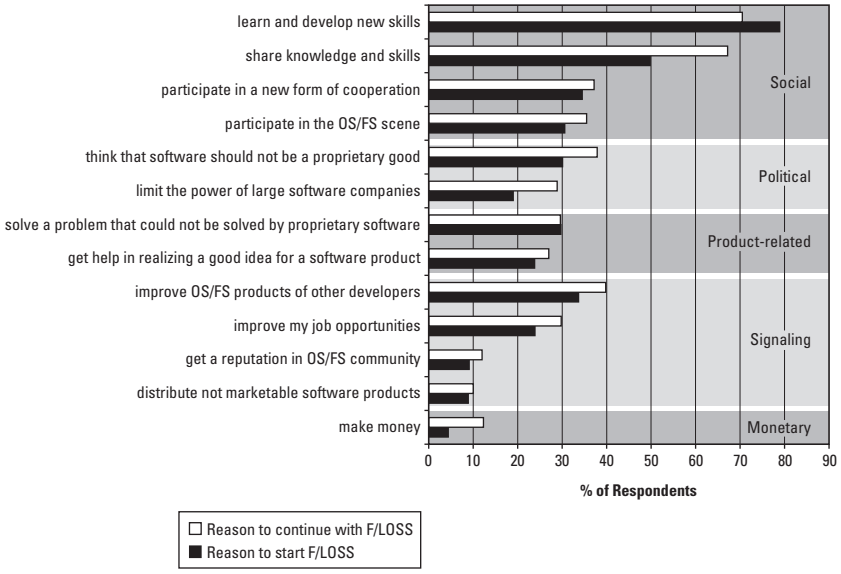

Figure 2.4
Initial and current motivations for FLOSS development (© 2002 International Institute of Infonomics)

the responses are consistent with self-interested participation and indicate that developers perceive a net positive value flow.

Figure 2.4 shows responses to the two main questions on motivation, asking developers for their reasons to first start developing free software and their reasons for continuing in the free software community. The chart groups the reported reasons into broad headings. Most notable is that the most important, reason to join and continue in the community is "to learn and develop new skills," highlighting the importance of free software as a voluntary training environment.

As a preliminary attempt to integrate the multiple, simultaneous reasons provided, respondents have been organized into four motivation classes (figure 2.5): social/community motives; career or monetary concerns; political motives; purely product-related motives. This is further explained in Ghosh et al. 2003, but in summary, while many developers express social or community-related motives, only those who also express career concerns were included in the second category, while only those who also express political views were placed in the third category. The last category is necessarily small, because it comprises those who expressed *only* product-

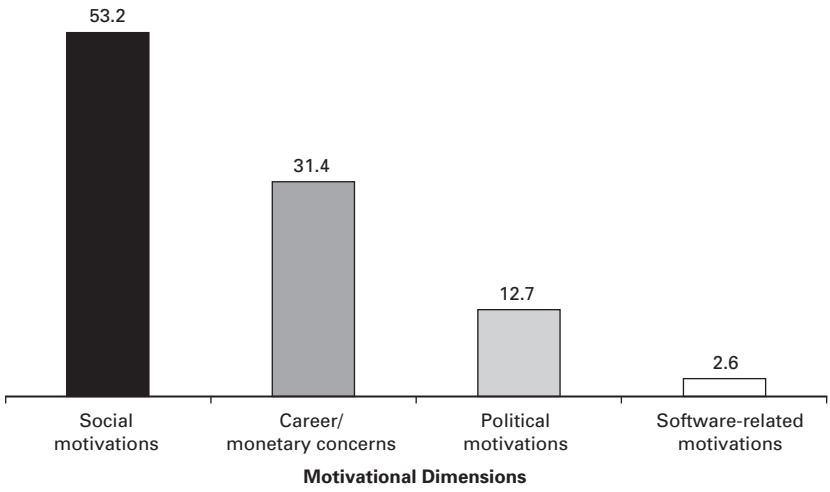

Figure 2.5
Developers by motive class, as percentage of total developers (© 2002 International Institute of Infonomics, FLOSS Developer Survey)

related motives (participating in the developer community to improve a software product, for instance). Of course, there are many ways of forming these categories and how one forms them changes how they correlate to other variables.

Organization

The FLOSS community shows simultaneously signs of both extreme concentration and widespread distribution. Measures of source code authorship show that a few individuals are responsible for disproportionately large fractions of the total code base and this concentration is increasing over time (the Gini coefficient for the Linux kernel[18] is 0.79). Several previous studies using a variety of objective metrics have shown that large fractions of code are developed by a small minority of contributors.[19]

However, the same studies and methods show that the majority of developers contribute relatively small amounts of code, and participate in a single or very few projects. Arguably, in the spirit of Raymond's (2001) view that "given enough eyeballs, all bugs are shallow" we can assume that the small high-contribution minority would not be as productive, and would be unable on their own to complete projects, without the support of vast armies of low-contribution participants. Moreover, the same objective metrics show that the majority of projects (or packages,

or even modules in the Linux kernel) have a single author or very few contributors.

Put together, these results suggest that the community is organized into not a single core with a vast periphery, but a collection of cores each with their own smaller, overlapping peripheries. This organization is fractal in nature, in that the shape of concentration curves[20] remains the same no matter what level of detail is used; that is, the distribution of high- and low-contribution developers is more or less the same whether one looks at "all projects on Sourceforge" or at "modules in the Linux kernel."

The fractal nature of this would suggest an organizational mechanism that is universal within the community, involving the development of strong leadership structures in a highly modularized environment. This is consistent with the FLOSS developer survey, where for the first time individuals were asked to measure their leadership role.

As figure 2.6 shows, only 35 percent of respondents claim to *not* be leaders of any project, while 32 percent report they lead a single project. Meanwhile, only 2 percent claim to lead more than five projects.

Figure 2.7 corroborates objective and other data sources showing that most people have been involved in a small number of projects (a similar

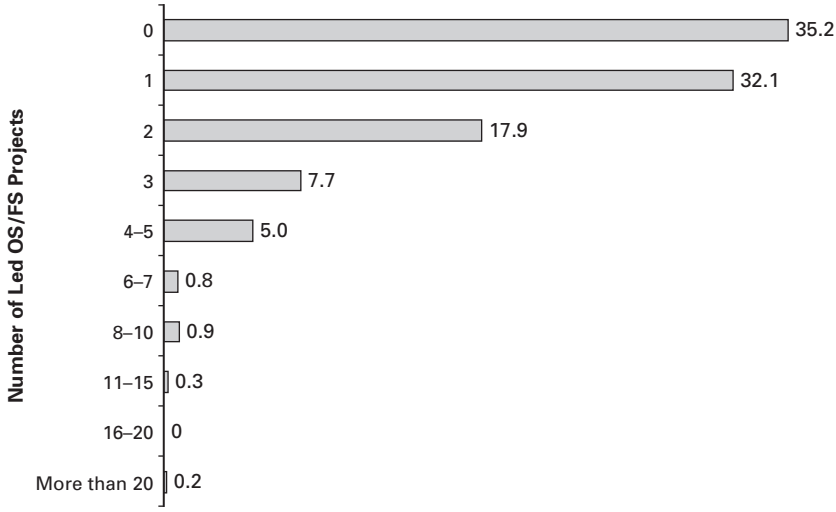

Figure 2.6
Leadership: number of projects involved in, percentage of respondents

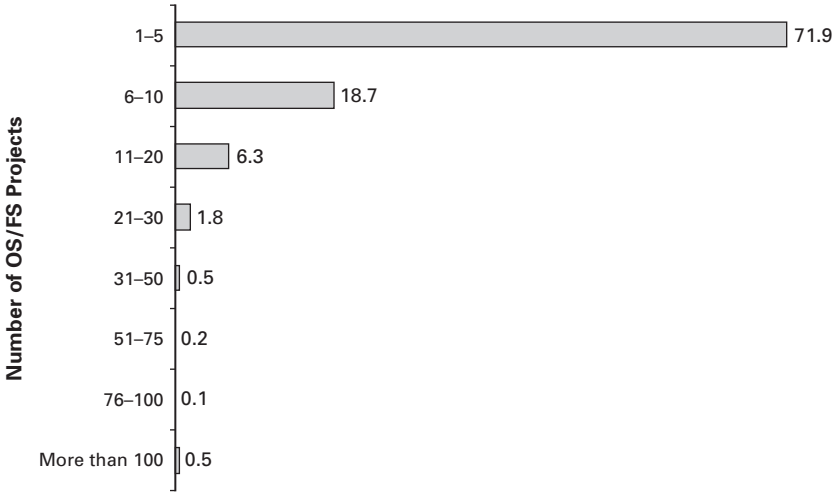

Figure 2.7
Number of projects involved in so far, percentage of respondents

but slightly different picture was reported when asked about current involvement in projects, which probably says more about a developer's ability to manage time than his degree of involvement as such).

Finally, most models assume that there is considerable collaboration and, more importantly, communication between developers in order to make this highly modularized, distributed form of production work. Figure 2.8 shows what people reported when asked for the number of other members of the developer community with whom they are in regular contact. The terms *contact* and *regular* were deliberately left undefined by us for two reasons: any definition we chose would be arbitrary, and we wanted to see whether developers believe they are in regular contact, which is an important insight into their own perception of the community's organizational structure. Surprisingly, as many as 17 percent reported being in regular contact with *no one*, thus indicating that significant development does take place in isolation (structural, if not necessarily social), at least between points of code release.

What is perhaps predictable, and consistent with the other findings, is that the vast majority of developers maintain regular contact with a small number of other developers—indeed, more than 50 percent are in contact with one to five others.

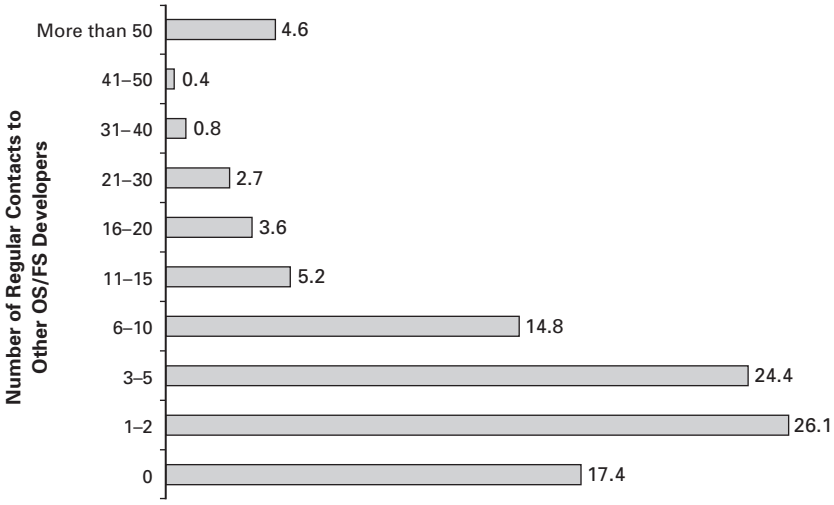

Figure 2.8
Regular contact with other developers, percentage of respondents

Subjective Responses, Objective Data?

Responses to any survey such as this one are necessarily subjective. Naturally, this is so even for the several questions that would have objective answers, such as the number of projects to which a developer has contributed, or monthly income, or whether such income is a direct result of involvement in free software development.

Some of these objective data can be checked against secondary sources, which may be more reliable or less subjective than survey responses. Some objective data, and even subjective data, can be checked against responses to other survey questions—although this doesn't make the responses any less subjective, one can at least know whether the subjectivity is internally consistent or self-contradictory.

Do Reported Motives Match Reported Rewards?

One of the most interesting things that can be checked for consistency is the relationship between reported motives and reported rewards. When asking people for their motives behind any action, one has to be aware that answers are not necessarily accurate, for several reasons:

- People are not conscious of their motives all the time.
- People might suppress some motives in preference for others, either unconsciously or deliberately, because of what they believe are the "correct" motives—this effect may be less striking in an anonymous Web survey than in a face-to-face interview, but probably still exists.
- People might report some motives but not others—especially when there are multiple motives, some that are "necessary but not sufficient" might be ignored.

This is certainly true for respondents to the FLOSS developer survey. But it is possible to get a truer understanding of respondents' motives by comparing reported motives to reported rewards. We assume that people who develop free software for a given motive (for example, to earn money) also report an above-average incidence of related rewards (above-average income). Not all motives have rewards that can be directly measured—most don't. Certainly, rewards generally result after a considerable time lapse, and such rewards would become apparent only through repeated panel surveys. But there is some degree of consistency control that can be performed within a single survey.

The first thing that comes to mind is whether there is any difference in reported income levels across motivation categories (as described previously), and whether motive category is reflected in earnings from participation in free software development. Figure 2.9 shows whether respondents from different motive classes earned income directly from FLOSS, indirectly only, or not at all. Figure 2.10 shows the mean income levels for motive classes (respondents were not asked for their level of income from FLOSS, so this reflects income from all sources).

Clearly, there is a strong indication that those who report career and monetary motives get what they want; that is, they report a low level of not earning income from FLOSS and high level of earning directly from FLOSS, as compared with other motive classes. In contrast, the politically motivated are least likely to earn any income, direct or indirect, from FLOSS. It is interesting to note that although those with purely product-related motives (such as distributing software that they could not market in a proprietary way, or looking at FLOSS as a method of implementing an idea) are less likely to earn money from FLOSS than other groups, they earn the most, on average (see figure 2.10). This is consistent with, as an example, a picture of software professionals with proprietary software as a (large) income source who turn to this development mode to solve a technical or product-related problem.

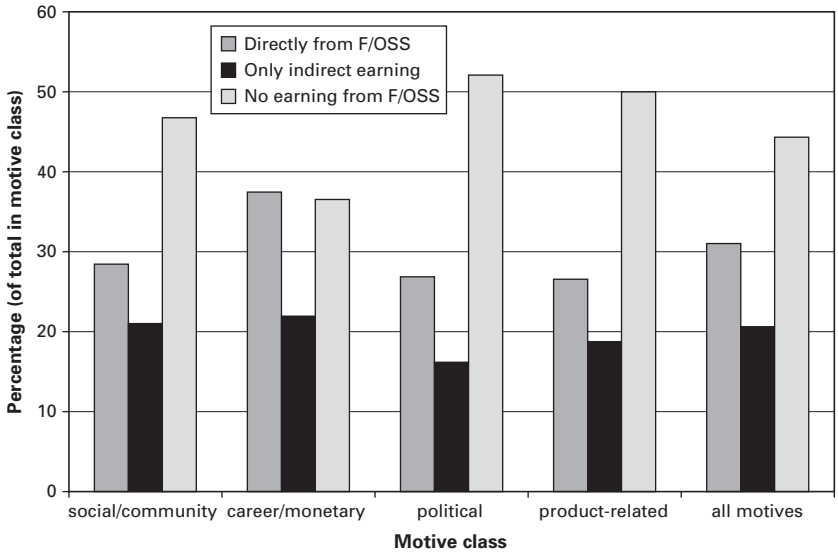

Figure 2.9
Earning from FLOSS by motive class

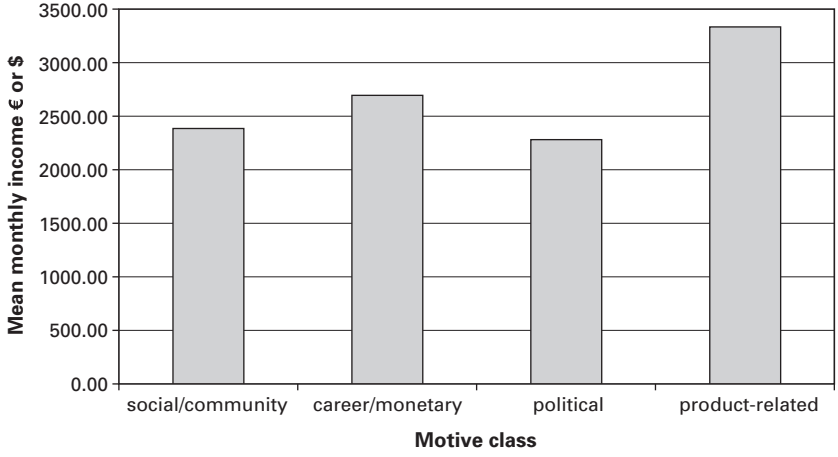

Figure 2.10
Income by motive class

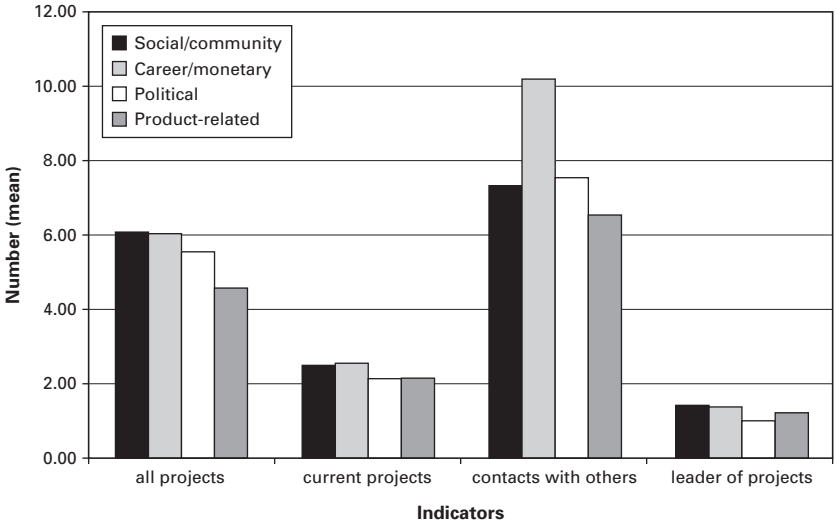

Figure 2.11
Social activity rewards by motive class

Figure 2.11 compares other possible reward indicators by motive class. These could possibly be treated as proxies for social or community-type rewards. While it seems clear that those motivated by career or monetary concerns get above-average levels of the appropriate sort of reward, the picture is blurred for the social/community and political motive class (at least from this set of indicators; there are many other candidates from the FLOSS survey). From this figure, it is apparent that social/community-driven and career/monetary-driven developers have similar levels of involvement in past and current projects as well as leadership of projects. (It should be emphasized that the career/monetary class includes those who did not explicitly state they wanted to earn money—those who were only interested in, for instance, reputation and possible job prospects form the largest share of this group.) What is striking is that the career/monetary group has a much higher level of regular contact with other developers than other groups do, possibly indicating that they are successful in developing the reputation that they value.

One "reward" indicator, in the sense that it sheds light on developers' perception of the sort of community to which they belong, is the Hobbes measure shown in figure 2.12. The Hobbes measure ranges from –1 (all others are altruistic) to +1 (all others are selfish). This indicator was

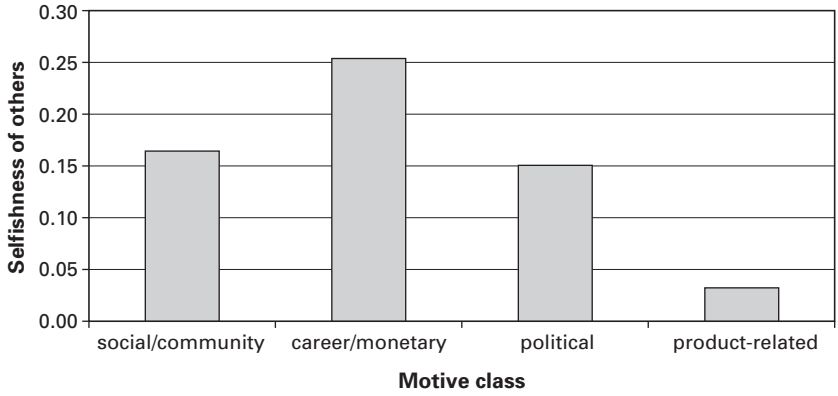

Figure 2.12
Others' selfishness—"Hobbes measure"—by motive class

calculated using a similar method to the "selfishness measure," in the form of the statement about other developers that "they give less than/more than/the same as they take" in relationship to the developer community. By and large, respondents were more likely to see other developers as altruistic and themselves as selfish. Unsurprisingly, the chart shows that a majority thought other developers in general are not altruistic, and notably more from the career/monetary group assumed that other developers are selfish than from any other group.

Developers motivated purely for product reasons have the most utopian view of the community, possibly because they feel for others, as they do for themselves (as the least "selfish" group, as seen previously), that there is more being put into the community than taken out of it.

Other Data Sources

It has already been shown for some variables that more objective data sources can corroborate the subjective data provided by respondents to the FLOSS developer survey. While a comparison of these data is beyond the scope of this text, the FLOSS source code scan (Ghosh et al. 2002, part V), the Orbiten Survey (Ghosh and Ved Prakash 2000) and the LICKS project (Ghosh and David 2003) provide extensive presentations of data on an aggregate level, especially on developer participation in and leadership of projects, that can be compared with subjective survey responses.

Conclusion

This chapter has presented some of the most interesting findings from the FLOSS developer survey. These findings highlight the crucial role that empirical data must play in the formation of models and hypotheses regarding the creation, organization, and activity of the free software developer community. The study also shows the utility of empirical data even when it is not possible to fully provide high levels of statistical accuracy, due to the unavailability of accurate data sources and census-type information on the universal population of developers.

Hopefully, this chapter provides the impetus for the process of using multiple sources of subjective and objective inputs on developer activity to better understand the motivations—and assess the rewards—behind the participation of individuals in this fascinating form of value production and exchange.

Notes

The FLOSS project was funded by the European Union's 5th Framework Programme (IST). The FLOSS consortium comprised the International Institute of Infonomics/University of Maastricht and Berlecon Research, Berlin. In addition to this author, Ruediger Glott, Bernhard Krieger, and Gregorio Robles were members of the Infonomics FLOSS team. Detailed results, the full questionnaire and analysis are in the FLOSS report (Ghosh et al. 2002, part IV).

1. This paper uses the terms *free software*, *open source software*, and *libre software* interchangeably, except where a specific distinction is made clear in the text. Most often the term *FLOSS* is used (Free/Libre/Open Source Software). Although it was originally a project title rather than a generic neutral term for the class of software, many commentators and writers have adopted FLOSS as such a term since the publication of the final FLOSS project report in July 2002. It should be noted that despite the media and policy focus on "Open Source" (in English, at least), *free software* is a more popular term among developers themselves—see http://flossproject.org/floss1/stats_1.htm.

2. Models and hypotheses are legion: "cooking-pot market" (Ghosh 1998a); "bazaar" (Raymond 2001); "gift exchange" (Barbrook 1998); "users turning producer" (von Hippel 2001a); "rational—signalling" (Lerner and Tirole 2002); "peer production" (Benkler 2002).

3. Such indicators may objectively answer at least part of the question about defining transactions: "who is doing how much of what with whom" seems to fall apart when the "how much" is no longer monetary, apparently eliminating the need for

actors to record their transactions. Nonmonetary indicators are described in more detail in Ghosh 2005 and a methodology for extracting them in an objective form from free software source code is detailed in Ghosh 2002.

4. A study of the distribution of code productivity among developers was first performed in the Orbiten survey (Ghosh and Ved Prakash 2000) by studying a sample of source code, and also using a different method based on annotations provided by developers in a large software archive (see Dempsey et al. 2002). Since then, statistics from developer portals such as Sourceforge.net provide at least the order of magnitude of the total developer population, if not accurate population size or demographic estimates.

5. In comparison to broader surveys (FLOSS; Robles-Martínez et al. 2001, which includes more than 5,000 respondents; Dempsey et al. 2002), the BCG survey (chap. 1, this volume) showed a higher degree of U.S. participants, more developers with several years of experience, and less animosity towards proprietary software companies. This may be related to the preselection criteria, which specifically included developers in mature projects on Sourceforge.net, a U.S.-based, open source portal. A sample that included, say, low-contribution developers on Savannah—a Free Software Foundation portal and hence more politically conscious—could have led to fairly different results.

6. See note 4.

7. Ghosh et al. 2002, part I. This survey of 1,452 industrial and public sector organizations in Europe on their use of open source software is beyond the scope of this paper. However, the FLOSS user survey did query organizations on their motivations for use, and also their support of developers, corroborating some of the results of the FLOSS developer survey.

8. Ghosh and Ved Prakash 2000; Ghosh 2002; Ghosh et al. 2002, part V; Ghosh and David 2003.

9. There have been small-scale studies of the organizational structure of the Apache, Jabber, Mozilla and other projects.

10. Ghosh and David 2003.

11. We assumed that developers can answer the survey in English, though—we didn't have the budget for a multilingual survey. As a result, we expect that we have an underrepresentation of east Asian and possibly Latin American developers.

12. See the FLOSS report, "Part IVa: Survey of Developers—Annexure on validation and methodology."

13. Ghosh 1994; Ghosh 1995; Ghosh 1998a.

14. Barbrook 1998.

15. Lerner and Tirole 2002.

16. Ghosh 2005.

17. Another way of looking at this is: if the selfish response is +1 and the altruistic response is –1 (the balanced response "I give as much as I take" is 0), then the "selfishness measure" is the mean of all responses.

18. In this context, the Gini coefficient measures the concentration of distribution: 0.0 represents uniform distribution (equal contribution from all participants), and 1.0 indicates full concentration (one participant contributes everything). The value here is for Linux kernel version 2.5.25, taken from the LICKS study of three versions of the kernel; see Ghosh and David 2003 for details.

19. The Orbiten survey (Ghosh and Ved Prakash 2000) and the FLOSS source code survey (Ghosh et al. 2002, part V) measured authorship of source code; Dempsey et al. 2002 analysed data from the Linux Software Map.

20. Lorenz curves, which plot authors' cumulative share of contribution and are the basis for Gini coefficients; for an example of their use in analyzing concentration of contribution in the Linux kernel, see Ghosh and David 2003.