Levels of Formality in FOSS Communities

Andrew Schofield[1], and Professor Grahame S. Cooper[2]

1 Salford Business School, University of Salford, Salford, M5 4WT, UK, a.j.schofield@pgt.salford.ac.uk,
WWW Home page: http://www.postgrad.isipartnership.net/~aschofield/

2 School of Computing, Science, and Engineering, University of Salford, Salford, M5 4WT, UK, g.s.cooper@salford.ac.uk,
WWW Home page:
http://www.cse.salford.ac.uk/profiles/profile.php?profile=G.S.Cooper

Abstract. One of the aspects of Free and Open Source Software (FOSS) which, in many cases, acts as the greatest deterrent to its adoption, is the method used to collaboratively develop the software and provide support through the use of communities. It is not until this method is examined more closely that its many advantages can be realised. The method can, however, seem very disorganised especially when compared with traditional proprietary development styles. A key difference between these two development approaches lies in the management of a project, and partly as a consequence, in the level of formality in the community environment. In terms of FOSS development, this usually entails the governance of not only the software development, but also of the community involved with it. These issues of formality and governance of communities and projects are the focal points of this paper. It presents the results of empirical survey research investigating FOSS community participants' views on the level of formality in FOSS, and the way in which this affects both development and support provision activities. The paper is then concluded by analysing what can be learnt from this examination of formality.

1 Introduction

Despite the many success stories and research studies demonstrating the advantages and potential of FOSS, there are still several barriers that deter individuals and organisations alike from using or developing it (Fiztgerald & Kenny 2003; Lakhani & Wolf 2003; Moody 2001, Pavlicek 2000). Apart from issues such as the economical model, and the lack of company backing, one of the major deterrents is the manner in which development and support activities are carried out.

The problem comes from the stereotypical view of FOSS code being thrown around within disorganised communities. In actual fact, these communities are arguably the most important element of FOSS. Much research has been done on FOSS communities (Ghosh et al 2002; Hann 2004, Hertel et al 2003; Krishnamurthy 2002; Lakhani & Wolf 2003; Oh & Jeon 2004; Pavlicek 2000; Raymond 2000; Scacchi et al 2005; Zhang & Storck 2001), revealing some interesting facts about them. Although it can be said that there is a general model that communities follow (Scacchi et al 2005; Schofield & Mitra 2005), it is also clear that all communities do not operate in the same way. Differences in working methods and styles of approach used become very apparent when comparing FOSS communities. An often noticeable difference is in the way communities are organised and governed. This is also seen as a major difference between FOSS and proprietary closed-source development (Raymond 2000; Pavlicek 2000; Moody 2001). FOSS development benefits from the large scale peer review approach available through open source code development. Proprietary closed-source development requires code to be kept secret in attempt to maintain a competitive advantage. Consequently, development teams are usually relatively small and often operate under fairly strict control. A hierarchical command structure is common in proprietary development, with developers answering to sub-team leaders, team leaders, department heads and so on.

This raises the interesting question of the effects of the working environments and governance, within FOSS communities. This paper attempts to answer this question by presenting the results of empirical survey based research, which collected FOSS community participants' experiences and views on the level of formality within FOSS communities. As these communities are often quite complex, the research was split into two sections, one focusing on the support aspects of the communities, and the other focusing on development activities.

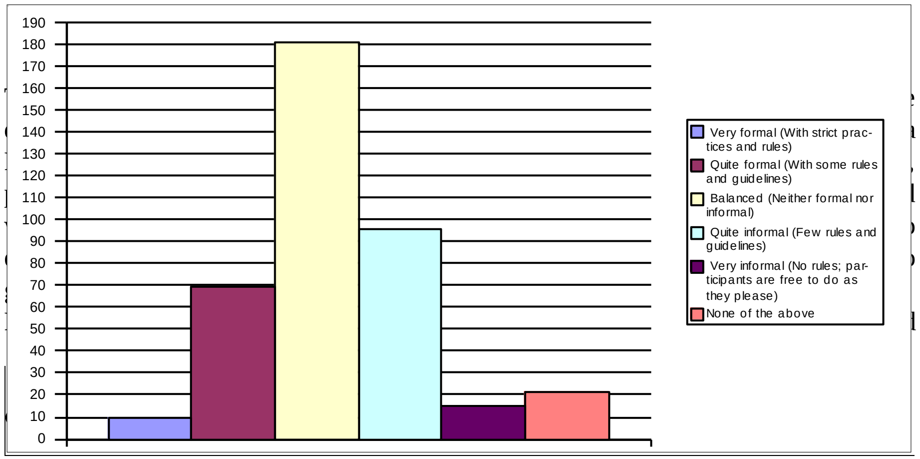
Research Method

As its primary data collection tool, the research used an on-line survey consisting of both open and closed questions to collect qualitative and quantitative data respectively. The survey technique was chosen because of its capability to reach a large number of research subjects.

The sample set for the research was the many Linux, BSD, and Open Source User and/or interest groups, hereafter referred to as Linux User Groups or LUGs. The research targeted LUGs within the UK, US, Italy, Germany and Canada, as these were the countries with the highest number of LUGs. In total 392 responses were received from the various countries.

It must also be pointed out that LUGs cannot be considered as absolutely representative of FOSS communities. However, members of LUGs are also, almost certainly, participants in other communities as well, either from the support or development side. The survey was also particularly designed to collect members' perspectives of FOSS communities in general. Additionally it should be noted that the sample consists of FOSS community members and although some may have proprietary software development experience, it should be recognized that a bias towards FOSS software will exist. The paper's purpose however is not to specifically compare the formality of FOSS and proprietary approaches, but rather to examine the effect of formality on FOSS.

Research Findings



FOSS communities to be 'neither formal or informal'. However, many of the additional comments left by those who chose this answer, stated that the formality of a FOSS community varies considerably between communities, and therefore it is difficult to generalise.

Interestingly however, 96 participants (24%) felt that they could generally describe FOSS communities as 'quite informal', and 69 participants (18%) described them as 'quite formal'. Very few participants chose either the 'very formal' or 'very informal' option, which received only 10 votes (3%), and 15 votes (4%) respectively.

There were many comments left for this question, these are grouped by topic and summarised below.

- Management: Some respondents pointed out that how formal a community is depends on how it is led, and that the formality of the methods used in a community project are needed only for project management purposes. Others wrote that the degree of formality is dependent on the maintainers, and the specific situation of the individual project. It was also stated that community based and company based software development often operate with the same rules, but the approach to how these rules are enforced differentiates the two types. To use the respondent's own words "*There are rules and guidelines, but they're not thrown in your face in the same way as they would be in a company".*

- Lack of Formality in Proprietary Software Development: Many respondents drew on their own experience, stating that proprietary software is often not developed in a formal or strict working environment. They felt that to position FOSS and proprietary software at opposite extremes of the 'formality scale' was incorrect and unrealistic, and that the image of proprietary software practices being very formal was just a myth, perhaps created as a form of propaganda by development companies. Some also stated that FOSS projects are often more formal than proprietary projects.

- Project/Community Dependant: A significant number of respondents wrote that the formality of a project or a community is very specific, and that a generalised statement which describes all of them is not possible.

- A Mixture of Elements: It was stated that communities can be viewed as informal in terms of them being open for anyone to participate. However, in terms of management of the final product, FOSS communities could be seen as very formal, as only the project leader decides what is incorporated into their design. Other respondents noted the informal practice of anyone being able to 'fork' the project and continue development in their own way.

- Informal Interaction: Although the interactivity side of FOSS communities may be fairly informal (i.e. discussion forums, etc.) the organisational aspects are a mixture of both formal and informal practices. Some felt that the formality of the interaction depends on how well one knows the project leader, and presumably the other community members.

- Geographical Dispersion: Some respondents stated that when working in a geographical local group, practices become more formal due to the added pressure of face to face meetings.

- Depends on the Projects' Complexity: Many respondents stated that the formality of a community depends on the project's size, scope, complexity, and maturity. Consequently, a large project, requires a high degree of formality to keeps things under control.

- Degrees of Granularity: Comments were left stating that for each of these large and formal projects, there are many smaller sub-projects which are far less formal.

- Theory Versus Practice: Several respondents with experience, wrote that although the working environment of proprietary development is usually very formal, it is only with respect to getting the job done, and less concerned is given to how it is done. This suggests that the formal aspects of FOSS development, such as the review of code submissions by the project leader(s), would have more of an effect on the quality of code, than the formal aspects of proprietary development, which may focus more on the task completion and meeting project deadlines. Many also wrote that in both the case of FOSS and proprietary software, formal rules and guidelines may be set down, but seldom followed. Several comments also referred to the use of versioning tools making many formal rules and guidelines unnecessary and superfluous.

- The Freedom of FOSS: Some respondents stated that FOSS is more informal because of its underlying ethos. They stated that those involved in FOSS do not want to be restricted to a formal and controlled system.

- Support is Often Informal: Several respondents stated that the support activities within FOSS communities are quite informal, but that the development side usually involves more formal practices.

Effects of Formality on FOSS Support Activities

Survey participants were then asked to suggest how formality affects FOSS support mechanisms. They were first asked to comment on whether they thought that the level of formality in FOSS communities (chosen in question 1) had a positive, negative, or no effect on the support aspects of FOSS communities (See figure 2).

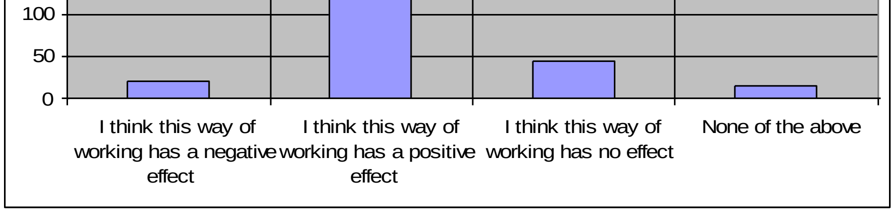


Figure 2: Effect of Formality of Support

Clearly most respondents (312 or 78%) believed that their chosen level of formality has a positive effect from a support point of view. Of those who chose this answer, 153 (49%) had chosen 'Neither formal nor informal' in the previous question. 'Quite formal' and 'Quite informal' received 57 (18%), and 75 (24%) respectively. The majority of those who felt there was a negative effect had chosen the 'Quite informal', and those who perceived no effect had chosen 'Neither formal nor informal'. However, it was the qualitative comments below that contain the most interesting results:

- A Deterrent to the use of FOSS: Respondents posted that if no formal rules or guidelines exist, then this may inhibit the acceptance of FOSS software. In addition some said that the unwritten rules or etiquette used in FOSS forums, could also make people feel unwelcome.

- Project Dependant: Several respondents stated that because of the varying degrees of formality in FOSS communities, the effects on support would also be varied. Some wrote that this depends on the size of the project. Large projects having good support because of the number of people involved, and smaller projects having poor support as the fewer members will have less time free to provide support.

- Formality Improves Support: Several respondents observed that communities with strict and formal working practices (FreeBSD was given as an example), have very good support, particularly documentation. Likewise, those less formal communities were often found to be lacking in support and had poor documentation.

- Arrogance Among the Knowledgeable: Some respondents felt that the informal nature of FOSS communities promoted a feeling of arrogance among the knowledgeable community members, who were then sometimes discouraging to 'newbies'. Clearly a commercial support system is more likely to be more formal and provide equal support to all levels of user.

- Ease of Using Forums: Other respondents felt that the informal nature of FOSS community forums makes it easier to ask questions. Contrary to the comments left above, it was suggested that because of the general equality and lack of a hierarchical system, members are more willing to help one another. The importance of interplay between experts and newbies was also emphasised.

- Direct Contact with Programmer(s): As a distinct advantage over the help-desk system, several participants wrote that FOSS communities allow direct communication to the actual writer of the code. This clearly has advantages as no-one could provide better support, and the questions asked might also stimulate ideas on further code development. In the proprietary world, non

Res
dev

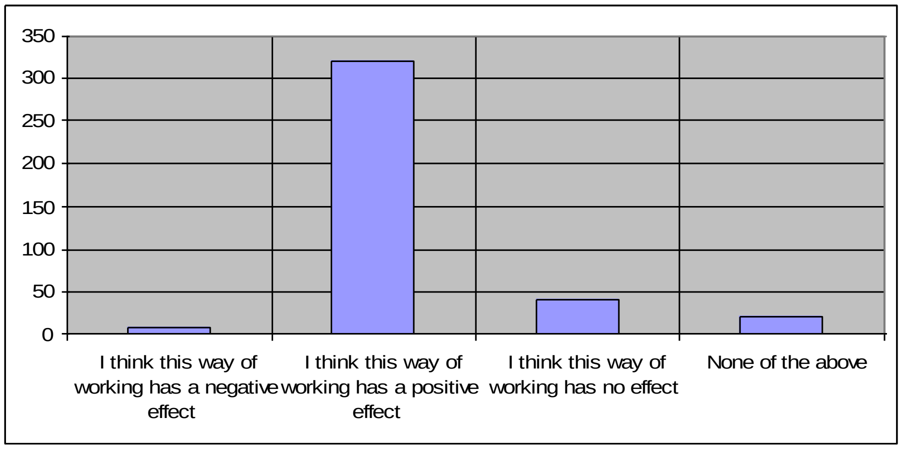


Figure 3: Effect of Formality on Development

As with the previous questions, the results are very conclusive: 321 respondents (82%) felt that their chosen degree of formality had a positive effect on development activities. Of these 321 respondents, 157 (49%) had previously selected 'Neither formal nor informal', 63 (20%) had chosen 'Quite formal', and 74 (23%) had chosen 'Quite informal'. The majority of those who felt there was a negative effect had chosen 'Quite informal'. The 'Neither formal nor informal' answer was mainly chosen by people believing this had no effect on development. However, only a few less participants felt FOSS communities were 'Quite informal' and that this also had no effect.

The respondents were once again asked to explain their answers and leave any additional comments that they felt were relevant. These are summarised below:

- The Big Picture: Without some control or steering group, respondents believed that programmers would do what they personally think is best. which may not be what is best for the overall project.

- Informal Management: Several examples were given by respondents of projects with an extremely relaxed approach. This *“hairy hippie style of management”* can lead to either unusable products or a forking of the project, leading to several very similar products.

- Natural Formality: Some respondents wrote of projects adapting and finding their own level of formality. One respondent referred to this as a community's *“natural formality”*. This means a specific level of formality would naturally evolve based on factors such as the participants, the size of the community, the project, etc. However, Too formal and developers will become irritated, too informal and a project's progress may be too slow.

- Development Feedback: Several survey respondents pointed out that the detection of bugs, and even design ideas can originate from questions asked on a support forum. Some described how design and development decision are usually made based on a consensual need, leading to a software evolution approach rather than large releases.

- Higher Formality for Larger Projects: Many respondents stated that they felt larger projects required a more formal structure to manage all the code submissions coming from the many different sources. However, other respondents felt that through the use of versioning software, this was not a problem. Some people also commented that larger projects need formal working practices to reduce the likelihood of project forking.

- Openness in Development: It was pointed out that the openness of code and approach can lead to arguments and disagreements, especially in very informal and undisciplined projects.

- Informal Development Rules: Although most respondents felt that at least some formal structure was required, some felt that the lack of rules was a good thing. For instance, some held the view that documenting slows down development, and therefore FOSS development was capable of releasing leading edge software quicker. Informal environments were also perceived as facilitating fast development, due to unrestricted communication. Several also felt that rules, such as documentation requirements, also discouraged some developers from participating. As one respondent wrote: *"Get the work done... someone will come along eventually to write it up".* Similar views were expressed about code writing styles. In many respondents' opinion, forcing programmers to adopt a specific style can be discouraging. Another viewpoint was that people do not get caught up in red tape and can concentrate on getting things done. As much FOSS development is also done in people's free time, some respondents felt that the informality of FOSS communities is a welcome change. Otherwise, in the words of one respondent *"it would just feel like work".*

- Informality Breeds Innovation: A general theme running through many of the submissions was that the freedom of an informal environment helps FOSS development to be dynamic and innovative. Some felt that a formal environment stifles creativity. Another stated advantage of an informal environment was that it allows developer to pursue an idea and follow it through without the pressure of it working at the end (i.e. experimental/trial and error approaches are possible).

- More control needed: Although it was a minority view, some felt more control was needed. They felt there was too much discussion through mailing lists about functionality and project direction. Another respondent wrote that FOSS development can often be *"too chaotic"* and that communities often have problems related to clashing egos and methods. Others felt that, rather than control, more planning is needed.

- Deterrents to Involvements: A disadvantage pointed out was that informal and open development environments would be a big shock to those who have previously working in a more structured environment. Others however felt that it would encourage participation but make it difficult for those core contributors who contribute the most.

- Communication Leads to Results: Simply being able to talk to each other was identified by several respondents as the key to good collaborative development. FOSS communities facilitate this communication and the informal approach makes it easy for developers to work together.

- The Developers Choice: Many respondents stated that the level of formality varies from community to community. Therefore most developers have the opportunity to choose to work on a project that has a level of formality that suits them.

- Formal Practices at the Right Time: Some respondents felt that both informal and formal approaches had their pros and cons, and that FOSS development should adopt the most suitable approach for a particular phase of the development cycle. Some stated that formal practices were not needed at the early stages of a project, but that as it grows, and more participants become involved, a *“benevolent dictator”* is needed. A few respondents referred to FOSS development as a Darwinist approach, becasue only the *'fittest'* submissions are accepted. Rather than a design plan and specification being drawn up and adhered to throughout the development, FOSS evolves and *“reacts to needs as they arise”.*

- Disadvantages of Voluntary Work: Although the fact that developers are largely volunteers makes them intrinsically motivated, some survey respondents felt that more effort is put into developing software that is fun to write, rather than the more mundane or boring applications. To quote one respondent *“that's why we only have 3 office suites but about 42,000 music players”.*

- Informality is Good for Growth: A few respondents wrote that although informal approaches are good for recruiting new members, they are often then less motivated to work than in a formal environment. Additionally, informal projects may grow quite fast but could subsequently stall or *“grow stale”*, from a lack of interest and participation.

Conclusions

The results collected allow us to further define the concept of formality with regards to FOSS. We can first separate formality into more specific important factors which we shall call 'managerial formality' and 'cultural formality'. Managerial formality refers to aspects of formality which are for the purposes of organisation and structure. These factors are concerned with the operational side of a project and manifest themselves as rules and regulations concerned with the support and particularly the development of FOSS. Many respondents wrote of the importance of formal management, particularly for large projects with many people involved and during phases of the development cycle where decisions about the direction of the development are made. Cultural formality refers to the level of formality which exists between the participants of the community. This is evident from the discussion boards and mailing lists of a community and is defined by the members themselves, with a possible influence from the managerial formality. This would also refer to the formality level which the survey participants described as a *“natural formality”*, and as evolving over time. The essential difference between the two is that one is imposed, or perhaps more accurately, suggested, by the community leaders, while the other comes into being or develops from the personalities and actions of the participants.

From the analysis of the responses, it seems that the predominant view is that managerial formality improves both support and development activities. Nevertheless, many survey participants warned of the dangers of an environment which was too formal. Likewise, cultural formality promotes freedom and innovation, but can be off-putting to 'newbies' or those used to formal practices.

To conclude, it seems that in the majority of cases, FOSS communities evolve their own natural level of formality, which may change over time depending on their members and the state of any projects. Although support and development activities are separable into two distinct areas of community, each with its own level of formality, the two exist in a symbiotic relationship whereby support is given based on the results of the development, and development itself is influenced by support activities. It is this informal mixing and matching that allows FOSS communities to function as they do.

References

Fitzgerald, B. and Kenny, T. (2003), “Open Source Software in the Trenches: Lessons from a Large-scale OSS Implementation”, Twenty-Fourth International Conference on Information Systems, Seattle, Washington, USA, December 14-17

Ghosh, R., A., Glott, R., Krieger, B., Robles, G. (2002), “Survey of Developers”, Free/Libre and Open Source Software: Study and Survey, International Institute of Infonomics, University of Maastricht, The Netherlands, Available at: http://www.infonomics.nl/FLOSS/report/ , (Accessed May 2006)

Hann, I. H., (2004) “Why Developers Participate in Open Source Software Projects: And Empirical Study”, Twenty-Fifth International Conference on Information Systems.

Hertel, G., Niedner, S., Herrmann, S. (2003), “Motivation of Software Developers in Open Source Projects: An Internet-based Survey of Contributors to the Linux Kernel”, Research Policy, Special Issue on Open Source Software Development, Available at: http://opensource.mit.edu/papers/hertel.pdf ,(Accessed February 2004)

Krishnamurthy, S., (2002) “Cave or Community? An Empirical Examination of 100 Mature Open Source Projects”, Available at http://opensource.mit.edu/, (Accessed Feb 2005)

Lakhani, K. R., Wolf, R.G. (2003), “Why Hackers Do What They Do: Understanding Motivation Effort in Free Open Source Software Projects”, MIT Sloan School of Management Working paper, Available at: http://freesoftware.mit.edu/papers/lakhaniwolf.pdf, Accessed (February 2004)

Oh, W., Jeon, S., (2004) "Membership Dynamics and Network Stability in the Open-Source Community: The Ising Perspective" Twenty-Fifth International Conference on Information Systems.

Pavlicek, R. C. (2000), "Embracing Insanity: Open Source Software Development", Sams Publishing, USA

Raymond, E. S, (2000), "The Cathedral and the Bazaar",
Available at: http://www.catb.org/~esr/writings/cathedral-bazaar/cathedral-bazaar/ (Accessed November 2003) and in Dibona et al (1999)

Scacchi, W., Feller, J., Fitzgerald, B., Hissam, S., Lakhani, K., (2005) "Understanding Free/Open Source Software Development Processes", Available at: (http://www.ics.uci.edu/~wscacchi/Papers/New/SPIP-FOSS-Intro-Dec2005.pdf)
(Accessed December 2005)

Schofield, A., Mitra, A. (2005), "Free and Open Source Software Communities as a Support Mechanism", UK Academy of Information Systems conference 2005, Newcastle, UK

Zhang, W. & Storck, J, (2001) "Peripheral Members in Online Communities",