**PPIG Newsletter – September 2006**

# Open source software communities: current issues

Françoise Détienne [(1)], Jean-Marie Burkhardt [(1,2),] Flore Barcellini [(1)]

(1) INRIA, Eiffel Group "Cognition and cooperation in design", Rocquencourt, France
(2) Université Paris 5, Laboratoire d'Ergonomie Informatique, Paris France

## 1. Introduction

Free Open Source Software (OSS) can be run, distributed, studied, changed and improved by its users thanks to a specific license (www.gnu.org) and its open source code. Furthermore, notwithstanding special concerns on distance effect on collaborative design and development (Détienne, 2006; Olson and Olson, 2000), OSS developers work in arbitrary locations and collaborate almost exclusively over the Internet, using tools such as email and software code tracking databases (e.g. CVS or Subversion). As such, OSS represents an extreme but successful case of geographically distributed design which challenges traditional software engineering practices as well as traditional co-located design.

The design behind OSS becomes an important phenomenon in the computer science world: there are thousands of OSS projects and millions of users of OSS systems, such as Linux, Apache or Python. The Free Open Source Software movement has received enormous attention in the last years. It is often characterized as a fundamentally new way to develop software that poses a serious challenge to the commercial software business dominating most software markets today .

It is claimed, for example, that defects are found and fixed very quickly because there are "many eyeballs looking for the problems" . Code is written with more care and creativity, because developers are working only on things for which they have a real passion. All these potential advantages are said to emerge from the following characteristics of work and collaboration inside OSS projects:

- OSS systems are built by potentially large numbers of volunteers.
- Work is not assigned: people undertake the work they choose to undertake.
- There is no explicit system-level design, or even detailed design.
- There is no project plan, schedule, or list of deliverables.

A part of the literature on OSS is primarily "ideological", i.e. it put forward ideas, goals and political arguments by OSS leaders and communities related to the development of OSS, in particular against the traditional software corporation that makes software a "black box". Beside this literature, empirical studies have started to address cognitive and performance issues related to OSS. Systematic studies have also been carried out in the field of sociology, management and also economy.

In this short review we aim at getting a picture on how OSS projects actually function and how it differs from, or conforms to the ideological picture. Our approach will focus on sociological and cognitive views and let aside management and economical issues well documented in the literature (see for example: Bonaccorsi and Rossi, 2003; Cohendet et al., 2000; Lerner and Tirole, 2002).

## 2. OSS : objectives and values shared within a community

OSS movement leaders (e.g. Raymond, 1999; Di Bona et al. 1999), as well as some scientific studies, often refer to OSS projects as being structured as communities (Bonnaccorsi and Rossi, 2003; Scacchi, 2001). More specifically, OSS projects can be seen as online communities according to Preece's definition (Preece, 2000): on the one hand, their members constitute a group of people connecting together on the Internet with a common goal which is to develop a software; on the other hand, the groups are led by the OSS ideology norms and principles (Elliott and Scacchi, in press; Stewart and Gosain, in press). Dalle and Jullien (2003) outline that, to be viable, a project has to reach a certain threshold depending on the number of participants involved, the strength of their relationships and their preferences for standardisation.

OSS projects can also be seen as epistemic communities (Cohendet et al. 2000; Conein, 2004). This means that the objective of OSS members is not only gaining individual knowledge but also co-constructing and sharing knowledge about the software they develop for the benefit of all the community. Two potential consequences are as follows. Firstly, participating in OSS development would foster the continuous learning of individuals in parallel to the dissemination of knowledge, practices and sources of innovation within the community. Secondly, users are directly integrated within the community, iteratively interacting with core developers of the OSS along the design process, and, as the source code is available, users that have the proper skill can adapt the software to some of their own needs. Thus, at least theoretically, the delivered software should have a better fit with their needs and a higher level of usability. These issues remain to be further investigated as some preliminary studies show, for example, that usability of OSS can be actually lower than expected from this analysis (Twidale and Nichols, 2005).

## 3. Who is participating in OSS?

The OSS survey and the OSS in Asia survey (Ghosh et al., 2002; Hiyane et al., 2004) highlight the average profile of OSS participants:

- They are mainly male (98%), less than 30 years old (75%), single or not living with someone (60%) and without children (83%).
- They are from all over the world, with a predilection for Europe (in particular France and Germany) and North America. Indians represent 1,9% of OSS developers of the surveys;
- They have a high level of education as 70% have an university degree and 83% are employed in the Information and Technology sector (33% are software engineers, 16% are student, 20% are programmers or consultant).
- For most participants, developing OSS is not their occupation and almost 70% do not spend more than 10 hours per week for developing OSS.

However, companies have recently been interested in OSS. Some companies sell services around OSS and apply OSS principles and way of work (e.g. Red Hat which is distributing Linux). Some OSS members may be partly sponsored by these service companies to participate as developers or as end-users but they can also be engaged by companies to work on OSS (e.g. the project leader of Python has been engaged by Google and works half-time for Google and half-time for Python).

## 4. What motivations for participating?

Motivations of software developers to engage in OSS projects have been outlined in recent surveys (Ghosh et al., 2002; Lakhani et al., 2002), and studies (Hertel et al. 2003; Lakhani and Von Hippel, 2003). Results suggest the following main motivations:

- To gain and improve their knowledge and skills;
- To gain reputation and honour from their peers. This point is also outlined by the OSS movement ideology (Raymond, 1999). Indeed, OSS projects are seen as meritocratic communities (Gacek and Arief, 2004; Mahendran, 2002; Crowston and Howison, 2005) i.e. technical expertise and individual contribution are seen as the way to socially evolve in the hierarchy within the community;
- To get benefit directly from their work. This point is related to the fact that developers are also users of the artefact they develop;
- To support the OSS movement ideology and beliefs that software have to be free and open source, and that the design process of OSS is better than the proprietary software one.

Finally working in OSS project provides prestige and visibility that give developers a chance to be noticed by software companies (Bonaccorsi and Rossi, 2003; Lerner and Tirole, 2002).

## 5. Some issues related to specificities of tools and working environments in OSS

It is usually assumed that OSS developers rarely meet face-to-face. Consequently, they coordinate their design activity almost exclusively in three information spaces (Barcellini et al., 2005; Ducheneaut, 2003; 2005; Sack et

al. 2003): the implementation space , the documentation space (Web Repository, CVS, man etc) , and the discussion space (discussion lists, forums, emails). OSS is thus an extreme case of distant design in which communication is mediated by tools and working environments. Two main categories of research issues are those of communication and coordination between members of OSS projects. Previous studies on distant design, in classical organizations, have identified several barriers to communication, leading to coordination breakdowns (Détienne, 2006; Herbsleb and Grinter, 1999; Herbsleb and Mockus, 2003; Krasner et al. 1987; Olson and Olson, 2000).

One research issue concerns whether there are communication barriers in OSS. One barrier, identified in the literature, refers to the lack of willingness to communicate, lack of trust and, in multi-site situations, to the poor perception people have to be part of the same team. This seems not to be the case in OSS projects in which participants feel as members of a community even if they do not meet together, except at some OSS conferences. Another barrier refers to the difficulty to establish contact. Knowing "who to contact about what" is recognized as problematic, in particular, in multi-site design. In OSS, participants seem to be aware of "who is the expert" and who to contact to get relevant information; if not, there are helped by other participants in the mailing lists. At last, a barrier refers to the culture of sharing and collaborating in an organisation (referred to as collaboration readiness by Olson and Olson) which is crucial in the adoption of tools to communicate. It seems that collaboration readiness is a key characteristic of OSS.

A second issue concerns coordination and how tools and practices in OSS support it. Mockus et al. (2002) assume that, for projects of more than 15 core developers, explicit coordination mechanisms are needed. Explicit design-related coordination mechanisms are elaborated, especially in large OSS projects (Mockus et al., 2002; Scacchi, 2001). The history of code and its different versions is supported by CVS or Subversion. These mechanisms are partially supported by tools. Bug report systems, such as in Bugzilla (Sandusky et al. 2004; Twidale and Nichols, 2005), are now provided by default with OSS repository platforms such as Sourceforge. However, there is a need to make clear how effectively and to which extent these tools support coordination activities in OSS design and whether they can be improved. Several authors (Ripoche and Sansonnet, in press; Sack et al. in press; Scacchi, 2001) stress that there is a real need to improve these tools in order to analyse automatically the huge amont of data available online (e-mails, bug reports, code and its versions) and make them more accessible and usable for developers.

Finally, it should be noted that the design process is not only done in a distant way but can also be done in co-located sites, e.g. Python sprints in which participants apply the XP and Agile methodologies to implement some chosen features of the software (Barcellini et al. 2006). How are these complementary ways of design involved and what factors could affect the performance and the evolution of the community are thus other interesting issues to be investigated.

## 6. Regulation in OSS

OSS design processes are not always as open-ended as the idealization might imply. Certain projects have prescribed means for controlling design and implementation contributions.
Implicit and explicit rules, inspired by the OSS movement beliefs and values (Elliott and Scacchi, in press) frame the activity within OSS communities, especially in the discussion space. For example, members have to be aware of the history of the project to avoid "spamming" mailing-lists with already-posted and answered topics; the risk is to be cautioned by other members and to fall into gaining reputation.

Some project went further by setting a prescribed process for proposing new software evolutions, for collecting community input on an evolution issue, and for documenting chosen design decisions. For example, it is the case for the Python Enhancement Proposal (PEP) set up in the Python project. In fact, the PEP process is quite similar to two design processes used in conventional software projects: Request For Comments (RFCs) and technical review meetings (D'Astous et al., 2004).

## 7. Social structure in OSS

**S**tudies of the social organization and the dynamics of design processes of specific OSS projects have shown that they often diverge from the idealized picture in a number of different ways. In particular, most major OSS communities have a strict, hierarchical organization that stratifies developers into levels (Gacek and Arief, 2004;

Mahendran, 2002). Centralized power structures of this sort are at odds with the flat, merit-based structure idealized by many OSS communities. Five different statuses are usually distinguished: the project leader - sometimes referred to (semi)-ironically as the BDFL (Benevolent Dictator For Life); the administrators (or core team); the developers or maintainers; the users.

Mahendran's ethnographic studies of Python's governance structure reveals a very conventional organizational structure: the project leader has control over the project; directly below him are a few people known as the Python Lab core team. Below them are members of a particular mailing list (Python-dev) who, like the core team, have the power to directly change the code of the project. Further down are advanced members who can comment on the project but cannot change the code, and finally newbies (or novices) at the bottom of the organizational hierarchy. Power is unevenly distributed across the discussion space, the implementation space and the documentation space (Sack et al. in press) that need to be mastered for participants to gain influence (Ducheneaut, 2005). Project participants with more power contribute to all of the spaces. Other participants with limited power have, literally, certain aspects of the project that are "off limits" to them. For example, not everyone can make changes to the code of the project. Mahendran concludes that the primary forms of power are material (who controls source code), discursive (who controls discussions), and technical (knowledge of and skill in programming). Control over these three sources is unevenly distributed across the project.

In a complementary way, the study by Barcellini et al (2005a; 2005b) provides a view on the cognitive activity in the discussion space and how these activities are influenced by the projects' social and organizational structure. For example, design discussions (PEP's) are focused and framed by specific members of the project, especially the project leader and the Champion : they initiate and maintain the design theme for discussion and lead the community to consensus. This study also outlines that there are real interactions between developers and administrators of the project along the design process. The authors also provide evidence that exchanges between OSS participants are mainly evaluative which is in accordance with the peer-reviewing process enhanced by OSS communities.

## 8. Users involvement in OSS

The users are considered as the main force of OSS design and development (Raymond, 1999). It is however important to note that in the OSS world, users are roughly of two types.

- They can be highly skilled in computer sciences even if their interest domains can be different (education, biology, scientific computing...). For instance, in the Python community, most users of the Python programming language are developers of modules based on Python, to satisfy specific needs in different application domains.
- End-users of these modules are specialists in the application domains, e.g. biologists in the case of Python.

OSS projects can be seen as participatory design as well as continuous design (Gasser et al. 2003). Whereas users, in traditional user-centred design models, take parts in the design process as informants - in the functional analysis phase- or as evaluators - in the prototype and simulation phases- in OSS projects, users could be more or less involved in different phases of the continuous design process.

A current issue concerns to which extent users' participation in OSS might be seen as similar to the participatory design methodology (Carroll, 1996). This participation is seen as one of the most important factor explaining the success and the quality of the designed OSS. However, end-users (i.e. not highly skilled in computer sciences) involvement in the OSS design process is probably quite marginal and requires boundary spanners, i.e. participants who transmit and translates needs between domains. End-users are probably those users that are referred to as passive users as they only use the software or lurk on the Internet (Preece et al. 2004). This gives rise to the risk that, as pointed out by Twidale and Nichols (2005), the OSS design process is not yet well armed to integrate software usability for their end-users.

Other users (i.e. computer-skilled) do participate to the evolution of OSS at several levels. Active users participate in mailing-lists discussions as informants for newcomers. They also mostly get involved in the debugging and maintenance processes (O shea and Exton, 2005), e.g., by reporting bugs and/or correct bugs with patches. They might also, more rarely, get involved in the design itself by pushing an idea for a new OSS

functionality: in this case they probably need to get support from other developers and/or from a member of the core team and to prove their implementation skills, as suggested by the preliminary results of a recent study (Barcellini et al. 2006). Further issues to be investigated are who are actually the users, how do they participate in OSS and what are the cognitive, social and technological factors affecting the design performance both in terms of technical soundness and in terms of usefulness, usability and reliability.

## 9. Conclusion

It is only recently that cognitive and social issues related to OSS have been empirically investigated. These studies have already provided a first picture of some of the sociological and cognitive issues on the process of designing and coding within these communities. The existing literature has been mostly focused on a few major and large projects, and it is not clear yet how these results may be relevant for the variety of OSS projects. Furthermore, many issues are still to be addressed as pointed out along this survey. Feedback from practitioners and from software companies that have tried to adapt OSS process for themselves are needed in complement to more systematic scientific studies as those presented in this survey.

## References

Barcellini, F., Détienne, F., Burkhardt, J-M., and Sack, W. (2005a) A study of on-line discussions in Open-Source Software Community: Reconstructing thematic coherence and argumentation from citation practices. In P. van den Besselaar, G. de Michelis, J. Preece and C. Simone (Eds), *Communities and Technologies 2005* (pp 301-320*)*. , Dordrecht, The Netherlands: Springer.

Barcellini, F, Détienne, F., Burkhardt, J.M., and Sack, W. (2005b) Thematic coherence and quotation practices in OSS design-oriented online discussions. In K. Schmidt, M. Pendergast, M. Ackerman, et G. Mark (Eds.) *Proceedings of the 2005 International ACM SIGGROUP conference on supporting group work* (pp 177-186). New York, USA: ACM Press.

Barcellini, F, Détienne, F., and Burkhardt, J.M. (2006) Users' participation to the design process in a Free Open Source Software online community. *Proceedings of PPIG'2006*, September 7-8, Brighton. UK.

Bonaccorsi, A., and Rossi, C. (2003) Why Open Source Software can succeed? Research Policy, 32, 1243-1258.

Carroll, J-M. (1996) Encoutering others: reciprocal openings in participatory design and user-centered design. *Human-Computer Interaction, 11*, 285-290.

Cohendet, P., Creplet, F. and Dupouët, O (2000) Organisational innovation, communities of practice and epistemic communities: the case of Linux. In A Kirman and Jean-Beboît Zimmermann (Eds) *Economics with Heterogeneous Interacting agents.* Lecture notes in Economics and Mathematical system. The Netherlands: Springer.

Conein, B (2004) *Communautés épistémiques et réseaux cognitifs : coopération et cognition distribuée* [web page] http://www.freescape.eu.org/biblio/rubrique.php3?id_rubrique=13, [20st of june 2005].

Crowston, K., and Howison, J. (2005) *The social structure of free and open source software developement.* First Monday [revue en ligne], 10(2), <http://www.firstmonday.org/issues/issue10_2/crowston>http://www.firstmonday.org/issues/issue10_2/crowston [référence du 06 juillet 2005].

D'Astous, P., Détienne, F., Visser, W., and Robillard, P. N. (2004) Changing our view on design evaluation meetings methodology : a study of software technical evaluation meetings. *Design Studies, 25*, 625-655.

Dalle, J.M., and Jullien, N. (2003) "Libre" software: turning fads into institution. *Research Policy, 32 (1)*, pp. 1-11.

Détienne, F. (2006) Collaborative design : managing task interdependencies and multiple perspectives.

*Interacting With Computers. 18(1),* 1-20.

DiBona, C., and Ockman, S. and Stone, M. (1999) *Open Sources: Voices from the Open Source Revolutio*n, O'Reilly and Associates Inc., Sebastol, CA.

Ducheneaut, N. (2003). *The reproduction of open source software communities.* Unpublished Ph.D. thesis, University of California, Berkeley, CA.

Ducheneaut, N. (2005). Socialization in an Open Source Software Community: A Socio-Technical Analysis. *Journal of Computer Supported Collaborative Work*, 14, 323-368..

Elliott, M., and Scacchi, W. (in press) Mobilization of Software Developers: The Free Software Movement. *Information, Technology and People.*

Fogel, K. (1999) *Open source development with CVS: Learn how to work with open source software*: The Coriolis Group.

Gacek, C., and Arief, B. (2004) The Many Meanings of Open Source. *IEEE Software, 21(1),* 34-40, January/February 2004.

Gasser, L., Scacchi, W., Ripoche, G., and Penne, B. (2003) Understanding continuous design in f/oss projects. In *Proceedings of the 16th international conference on software engineering and its applications (icssea-03)*. Paris, France.

Ghosh, R.A, Glott, R., Krieger, B., and Robles, G. (2002). *Free/Libre and Open Source Software: Survey and Study*. FLOSS Deliverable D18: FINAL REPORT. Part 4: Survey of Developers. International Institute of Infonomics, University of Maastricht, The Netherlands , June 2002.

Herbsleb, J.D., and Grinter, R.E. (1999) Splitting the organization and integrating the code: Conway's Law revisited. *International Conference on Software Engineering.* 1999, Los Angeles, CA, p 85-95.

Herbsleb, J.D., and Mockus, A. (2003) An empirical study of speed and communication in globally-distributed software development. *IEEE Transactions on Software Engineering, 29(6).*

Hertel, G., Niedner, S., and Herrmann, S. (2003). Motivation of software developers in Open Source projects : an Internet-based survey of contributors to the Linux kernel. *Research policy, 32*, 1159-1177.

Hiyane, K., Iio, J., and Shimizu, H. (2004) *Free/Libre/Open Source Software Asian Developers Online Survey (FLOSS-ASIA).* March 16, 2004. Mitsubishi Research Institute, Inc. Available at http://oss.mri.co.jp/floss-asia (June 7, 2006).

Krasner, H., Curtis, B., and Iscoe, N. (1987) Communication breakdowns and boundary spanning activities on large programming projects. In: Olson, G. M. Sheppard, S. Soloway, E. (Eds.), *Empirical Studies of programmers: second workshop*, Ablex. pp 47-64.

Lakhani, K. R., and Von Hippel, E. (2003) How open source software works : « free » user-to-user assistance. *Research Policy, 32,* 923-943.

Lakhani, K., Wolf, B., and Bates, J. (2002) *BCG Kacker survey*. The Boston Consulting Group, In Cooperation with OSDN, January 31, 2002. Available at www.ostg.com/bcg/BCGHACKERSURVEY.pdf (June 7, 2006).

Lerner, J., and Tirole, Jean (2002) Some Simple Economics of Open Source. *Journal of Industrial Economics, Vol. 50, No. 2, pp.* 197-234.

Mahendran, D. (2002) *Serpents and Primitives: An ethnographic excursion into an Open Source community*.

Master's Thesis, School of Information Management and Systems, University of California at Berkeley.

Martin, F. (2002) Les forums électroniques: activités de communication ou de production? In E. Engrand, S. Lambolez, A. Trognon (Eds), *Communications en situation de travail à distance* (pp.183-193). Nancy, France : Presse Universitaire de Nancy, collection langage-cognition-interaction.

Mockus, A., Fielding, R.T., and Herbsleb, J. (2000) A Case Study of Open Source Software Development: The Apache Server. In proceedings, *International Conference on Software Engineering*, pages 263-272, Limerick Ireland, June 5-7.

Olson, G.M., and Olson, J.S. (2000) Distance matters. *Human-Computer Interaction, 15*, 139-178.

O'Shea, P., and Exton, P. (2005) The Role of source code within program summaries describing maintenance activities. In P. Romero, J. Good, E. Acosta Chaparro and S. Bryant (Eds) *Proceedings of PPIG 17*, pp160-172.

Preece, J. (2000) *Online Communities: Designing Usability, Supporting Sociability. Chichester*, UK: John Wiley & Sons. (439 pages)

Preece, J., Nonnecke, B., and Andrews, D. (2004) The top five reasons for lurking: improving community experience for everyone. *Computer in Human behavior,20*, 201-223.

Raymond, E. S. (1999) *The cathedral and the bazaar.* Available at **http://www.tuxedo.org/esr/writings/cathedral-bazaar/** [June 20, 2005].

Raymond, E. S., and Young, B. (2001). *The cathedral and the bazaar: Musings on linux and open source by an accidental revolutionary*: O'Reilly & Associates.

Ripoche, G., and Sansonnet, J-P. (in press) Experiences in automating the analysis of linguistic interactions for the study of distributed collective. *Journal of Computer Supported Collaborative Work.*

Sack, W., Détienne, F., Ducheneaut, N., Burkhardt, J-M., Mahendran, D., and Barcellini, F. (in press) A methodological framework for socio-cognitive analyses of collaborative design of Open Source Software. *Journal of Computer Supported Collaborative Work.*

Sack, W., Ducheneaut, N., Mahendran, D., Detienne, F., and Burkhardt, J.-M. (2003) Social architecture and technological determinism in open source software development. Paper presented at the *International 4S Conference: Social Studies of Science and Society*, Atlanta, GA.

Sandusky, R.J, Gasser, L., and Ripoche G. (2004) Information practices as an object of DCP research. Paper presented at the Distributed Collective Practices workshop in *CSCW'04*. November 6-10, Chicago, US

Scacchi, W. (2001) Understanding the requirements for developing Open Source Software Systems. *IEEE Proceedings--Software, 149(1),* 24-39

Stewart, K. J., and Gosain S. (in press) The impact of ideology on effectiveness in Open Source software development teams. *MIS Quarterly.*

Twidale, M.B., and Nichols, D.M. (2005) Exploring usability discussions in Open Source development. In Proceedings of *HICSS '05*, pp198c- 198c.