A Participation Paradox: Seeking the Missing Link between Free/Open Source Software and Participatory Design

Zegaye S. Wubishet
University of Oslo, Norway
Email: zegayeseifuw@yahoo.com

Bendik Bygstad
Norwegian School of Information Technology, Norway
Email: bygben@nith.no

Prodromos Tsiavos
London School of Economic, UK
Email: p.tsiavos@lse.ac.uk

***Abstract*—The success of Free Open Source Software (FOSS) has resulted in thousands of robust and ubiquitous products such as Linux, Firefox and Apache. However, the usability of many other FOSS products is often poor, and the most successful projects are the ones where the user and the developer are one and the same. The lack of broader participation is worrying, because it threatens the entire production model of FOSS. In this paper we investigate the reasons for this situation, drawing extensively from research on participatory design and commons based peer production (CBPP), and on a case study of three FOSS projects. Potential lessons are also drawn from the CBPP model in general, and the FOSS approach in particular, to mitigate the challenges facing distributed participatory design (DPD).**

***Index Terms*— open source software, participatory design, commons based peer production**

I. Introduction

The success of Free Open Source Software (FOSS) has resulted in thousands of robust and ubiquitous products such as Linux, Firefox, and Apache. However, while open source software has generally proven to be useful and reliable, usability problems for non-expert users have been consistently reported [2, 12]. The most successful FOSS projects are the ones whose users are also developers, but the non-technical user may feel that FOSS software is developed by experts for experts.

Although user involvement has been emphasised as an important element in FOSS development [40], low participation of end users is salient; developers usually rely on their own intuition to obtain requirements or learn about users indirectly [25]; project leaders and core members exert more influence [53]; and in most cases the developers produce the systems for their own use without considering the particular needs of the non-technical users [29, 53].

It is reasonable to assume that such scarce participation of end-users strongly affects the usability and success of FOSS systems; the problem is believed to be one of the key barriers to the diffusion of FOSS to desktops of ordinary users [40]. There are, however, even deeper practical and ideological concerns. If participation does not become wider and more substantive, it may threaten the entire production model of FOSS.

The consequences could be that (1) the economic model behind FOSS collapses and its benefits are never materialised, (2) the power relationships that underlie the FOSS model collapse; instead of reversing power structures and increasing the autonomy of the user and developer it intensifies the existing power regime and (3) in a wider perspective, this results in under-utilization of the FOSS commons and the promised freedoms by the movement, implying the traditional approach is sufficient for end-users' purposes.

Therefore, it seems paradoxical that the very same possibilities of participation that FOSS culture and practice provide impede the participation of lay users. In addressing these issues, we ask the following questions:

- Why do FOSS systems suffer from scarce end-user participation and therefore from poor usability?
- What lessons can be taken from the Participatory Design (PD) tradition to create a better user experience?
- What can the FOSS model offer to PD?

In order to answer these questions, we will deepen our investigation and ask whether features of the production model behind FOSS are responsible and/or helpful for this situation. We need to return to a FOSS abstraction, such as the Commons Based Peer Production (CBPP) model and interrogate some of its premises to ascertain whether they invite or prohibit participation after a certain point. Drawing on the CBPP model, this study reveals how the tradition of PD could respond to these practical and power concerns in FOSS. Potential lessons from FOSS and CBPP are also linked to the PD tradition.

Even though user participation is a much studied concept in the information systems literature, only a few researchers have examined the problems related to the participation of non-developer users in the FOSS development context [29]. There has been a longstanding but unmet need to extend the benefits of the PD approach to the present reality of distributed production [26, 34, 41]. This study argues that the fundamentals of the CBPP model [10] are the bridging links of both traditions, empowering end users and maximising participation. We believe that this way of re-thinking the issues will contribute specifically to the FOSS literature and more generally to the IS research field.

The next section reviews the research, exposing the challenges and identifying possible contributions from PD and CBPP. This is followed by a discussion on the methods. In section 4 we analyse the relationships among the three strands of research. In section 5 we empirically discuss how PD and CBPP might strengthen the FOSS community, and conclusions follow in section 6.

II. Review of Related Literature

In this section we review the FOSS research, and relate it to participatory design and CBPP research. This comparative analysis identifies the challenges and relationships of the three approaches. It also provides an analytical lens through which we analyze the cases under discussion.

A. *FOSS*

The essence of free and open source software is that the source code is released along with the software to anyone who chooses to use it. The code is open, public, and non-proprietary [52]. Anyone in possession of the software has the freedom to run it for any purpose, to study how it works, adapt it to his/her own needs, to redistribute copies to others, and to improve the program, and share the improvements with the community [47].

The core institutional device ensuring these rights is the licensing mechanism which inverts the idea of exclusion as a basis of property rights; it offers every individual the right to distribute, but not the right to exclude [52]. The development usually takes place asynchronically in a distributed manner [13]. In the open source community it is strongly believed that if users are properly cultivated, they can become co-developers. Given a bit of encouragement, the users will diagnose problems, suggest fixes, and improve the code far more quickly [44].

The often-cited success examples of FOSS are the Linux operating system, the Apache web server, the Mozilla browser, the GNU C compiler, the Perl scripting language, and MySQL database management system. One of the key factors in the success of such systems is that the development process thrives on increasing the user and developer base [45]. In fact, most of these systems are infrastructural, benefiting from a larger pool of interested participants, and their requirements are part of the general taken-for-granted wisdom of the software development community [21].

At the same time, many other FOSS projects fail to attract developers and/or users and, as a result, never get off the ground [43]. The majority of FOSS projects located on popular hosting sites such as SourceForge only have a few members [17, 57]. Although FOSS over the last decade has moved from the hacker margins to the mainstream [52], opinions differ about the true participative nature of the process [54]. In practice, up to 90 per cent of the potential users might be 'passive users' who merely use a system and take no part in its development [40]. This process results in software developed by experts for experts [54]. Such lack of average user participation poses a serious challenge to the usability of FOSS systems [39].

Implied causes of scarce involvement of non-technical users in the literature include the following: users do not have the technical vocabulary valued by developers so that they will be rarely attended [40]; developers have a limited understanding of usability and there is a lack of resources and evaluation methods fitting into the FOSS paradigm [2]; requirements are taken as generally understood and not needing interaction among end-users [21]; weak focus on integration of usability concepts and approaches with software development methodolgies [14]; and some designers and developers have customarily viewed software as a technological problem rather than as a people problem [20, 25, 54].

The tools of participation, such as Concurrent Versioning Systems, Mailing Lists or Electronic Fora may add to the confusion among end-users who wish for a simple yet powerful set of tools. Thus, non-expert users could be intimidated by the ability required to fully customise FOSS software or even perceive the possibility of participating to the production of source code as a sign of extra complexity and a source of confusion [21].

Power structure is also an issue in FOSS literature. The open source software development process is not a free-for-all where everyone has equal power and influence [52]. The technically capable and active core team members usually have more authority and decision making rights [29]. The criticism and feedback from developers is taken much more seriously than that of end-users [35]. Even among the developers themselves, there is politics in the management of conflict where there are manipulations of power, interests, rules, behavioural norms, decision-making procedures, and sanctioning mechanisms [52].

B. *Participatory Design (PD)*

Participatory Design is a set of theories and practices considering end-users as full participants in software and hardware production activities [24, 46]. As a design philosophy it recognises the critical importance of people within information systems development [22] and views organisational issues as central to system design [15].

Originating in the Scandinavian workplace labour movement [22], the goal of PD was to democratise work environments and increase job satisfaction [19, 41]. Later it developed to encompass issues of quality and better acceptance of information systems [34]. It assumes users to be experts in their work, and able to design and develop the tools they use [5, 46]. The users' involvement yields better requirements specifications, and results in better system design and more usable software [41]. PD focuses on the direct participation of those who will be affected by the development of the application in the decision-making, design and development stages [50].

At the core of the democratic and participatory objective is the concept of user/worker 'empowerment' [16, 41]. Such empowerment can be democratic, which maintains that users have the right to participate in decisions, or functional, by which users have the right to be able to perform their job effectively and efficiently, and their participation in the design process is needed in order to achieve this [30].

A main principle in PD is a mutual learning process between developers and users in an organisational context [11]. PD stresses that developers need knowledge of the actual use context and users need knowledge of possible technological options [26, 46]. Its epistemological stance is that such types of knowledge are developed most effectively through active cooperation [31], which is a core issue in FOSS projects.

Although PD is a well-known and quite commonly used approach, no formalised guidelines applying it to information systems development have as yet been established [42]. Hence, techniques and methods and their utilisation vary from project to project. However, apart from the political and theoretical explorations of participation, PD researchers have developed practices that promote cooperation [31].

One popular technique in traditional PD is conducting workshop sessions between users and developers. This assists in information sharing; users learn design skills and understand technological possibilities, and developers understand the organisation and attitudes towards work. Other techniques and tools focussing on systems design are brainstorming, scenarios, mock-ups, simulations of the relation between work and technology, future workshops, design games, case-based prototyping, and co-operative prototyping [31]. Surveys are also used to evaluate usage in different contexts and by different user groups [27].

Envisionment is also central to PD which is experiencing the system functioning in the use situation. With this technique, the user works under the conditions that the proposed system will bring about; this therefore requires more than reading a description of the proposed system or watching a demonstration [15].

However, a fundamental limitation of PD techniques is that they are primarily focused on project stakeholders being collocated [27, 31], and on the development of a single, contiguous, customised software system representing and supporting workflows within one organisation [41]. This is fiercely challenged in the face of the growing trend towards distributed systems development. New organisational structures such as virtual networks are emerging, where formal organisational structures are missing and boundaries between stakeholders are becoming more fluid [41]. These contexts may set practical limits to the applicability of traditional PD techniques.

Recently, a trend towards Distributed Participatory Design (DPD) has emerged in the literature. DPD recognises that many contemporary design teams often comprise developers and users that are geographically dispersed [34]. Obendorf et al [41] stressed that 'to cope with other use contexts and new forms of work – such as communities and virtual networks – the traditional repertoire of PD methodology needs to be expanded to deal with distribution and diversification of users'.

According to Gumm [26], the distribution could be in terms of varying locations of people and resources (physical), related to work structures (organisational), or variations in working hours (temporal). The physical distribution is the bigger challenge for PD and the main area of potential conflict [26]. DPD intends to cope with such settings where stakeholders are distributed across time, space, and/or organisation [26, 41].Along the needs of the distributed development trend, PD is understood to include understanding, designing and evaluating activities [28]. The aim of the activities is to improve the functionality and/or usability of the solution. This can also involve different kinds of intermediaries in the process.

Still, a major challenge of DPD is that the concepts of real participation and of physical distribution tend to be in conflict as most PD approaches are based on the possibility of face-to-face meetings [27, 34]. The challenge emanates from the reasons that distributed teams have multiple differences such as language, experience, technical and domain knowledge, and time zones [37].

In response to the challenges facing DPD, Obendorf et al [41] introduced two techniques extending those known in the traditional PD practice: inter-contextual user workshops and commented-case-studies. Inter-contextual user workshops bring users from different contexts and developers together to reflect on the usage and design of the software and its future development. Commented-case-study makes the face-to-face interaction of the inter-contextual workshop more persistent by providing a written documentation of use experiences and design decisions.

In contrast to more traditional PD workshops, inter-contextual user workshops reflect and enhance use practices within the current system design rather than designing new or revised features. Unlike moderated focus group mechanisms, these are meant to be oriented more towards compromise, consensus and mutual learning [41].

Loebbecke and Powell [34] argued that as PD tackles the problems that arise in distributed projects, it is more necessary to look outside its own domain for solutions.

They demonstrated how other collaborative research methods, like action research and design science, may offer useful experiences. Accordingly, this paper seeks insights from the traditions of FOSS and CBPP to enrich the PD approach, particularly as it applies to Information Systems field. The next section introduces CBPP as a potential linking phenomenon.

C. Commons Based Peer Production (CBPP)

CBPP is a new model of economic production where the coordinating mechanism is neither market nor managerial hierarchy, but a digitally networked environment [9, 10]. Examples of cases that manifest this model include content development (Wikipedia, NASA Clickworkers experiment), relevance and accreditation purposes (Amazon, Google, open directory project, Slashdot, Kuro5hin), distributed services (Seti@home, Gnutella, distributed proofreading), and free and open source development (GNU/Linux, Apache web server Mozilla and others) [10].

Benkler contends that autonomy, democracy, justice and development can all be improved with peer production, which implies a shift from consumers to users, doing more for and by themselves, and in a loose association with others [9].

The critical mass of participants in such peer production projects do not generally participate, for financial reasons. According to Benkler, this form of production has two core characteristics. The first is decentralisation where authority to act rests with participants, rather than a central organiser or manager. The other is that it uses social cues and motivations, rather than prices or commands, to motivate and coordinate the action of participating agents.

Benkler [8, 9, 10] proposes three characteristics of successful peer production and participation:

- the project and artefact must be modular
- the modules should be predominately fine-grained
- there should be low-cost integration mechanisms

These properties determine the number of participants, the scope of varied investments (heterogeneity), the minimal investment required to participate, and the simplicity of integration. Benkler argued that a project with these features can attract many users – both technically good contributors and end-users. The modularized approach also helps to reduce the learning curve for newcomers [21]. Moreover, he stressed that not every chunk needs to be fine grained.

In an ideal CBPP scenario the modules are granular and heterogeneous, so that individuals with even minimal skills and time could self-select themselves to participate in such a project. However, recent studies indicate that in the course of a project's life-cycle the capacity to participate required from the peers increases, so their participation is constantly under pressure [10, 49, 51] . We will return to this premise with reference to PD approaches in the discussion section.

Another key element of Benkler's CBPP model is that of excess capacity, both at the level of the artefact and that of the peer. Excess capacity at the level of the artefact relates to the processing, storage and communication technologies which are ubiquitously available [10]. On closer inspection, however, it also relates to the non-rivalerous nature of the artefact that is to be produced; the use of the produced artefact by one user should not hinder the enjoyment of another. This feature of the produced artefact allows maximum dissemination and parallel production and hence contributes to the increasing of the user participation. However, it needs to be complemented by 'peer excess capacity' [51], by which whoever participates in the development needs to have the skills and the time to do so, and these skills and time need to persist over time.

Peer-production processes generally also require some substantive cooperation among users, including, in FOSS development, spotting a bug, proposing a fix, reviewing the proposed fix, and integrating it into the software are interdependent acts that require cooperation. This is accomplished by a combination of measures such as technical architecture, social norms, legal rules, and a technically backed hierarchy that is validated by social norms [51]. Such structural, philosophical and moral opportunity prevalent in FOSS projects, is attracting adn resulting a shift of thinking in commercial companies as well [1].

According to Benkler, in the traditional mass-media model ownership of the means of communication provides an owner the power to select what others view, and thereby enables it to affect their perceptions of what they can and cannot do. The networked information economy, however, provides varied alternative platforms for communication (like mailing lists and dynamic pages) that can moderate this inordinate power. Benkler explains that the Free Software model in particular has shown us that successful peer-production projects can be technically and culturally structured in ways that make it possible for many individuals to contribute vastly at levels of effort that are commensurate with their ability, motivation, and availability.

III..ANALYSIS

This section gives a theoretical and comparative analysis of how the FOSS, PD and CBPP models are related in view of addressing the posed problems. The section starts with a summary table (Table 1) from the discussion so far, highlighting the core attributes of the three approaches.

A. Participation in FOSS and PD/DPD

According to the critical review in Section 3, it is shown that widening the participation and building a community of users is both a success factor and an ideological issue in FOSS development. Here we contend that many of the acceptability and usability problems that plague traditionally engineered systems in general, and a large number of FOSS projects in particular, can be mitigated by adopting the PD techniques.The practices

TABLE 1.
SUMMARIZING THE ATTRIBUTES OF FOSS, PD AND CBPP

	Free and Open Source Software	**Participatory Design**	**Commons Based Peer Production**
Focus	Open and participatory development; freedom to study, use, improve and redistribute software	Workplace improvement, user empowerment: both democratic and functional	Social production: decentralised and based on social cues rather than managerial commands or market-based prices
Relationship to users	Users are developers and developers are users	Developers and users should learn from each other	Users contribute directly to production through an open IT architecture; exploiting excess capacity
Key Principles	Peer-to-peer development	User-developer co-operation: • Workshops • Brainstorming • Prototyping	IT architecture: • Modularity • Granularity • Low-cost integration

there can also engender a sense of ownership, giving users a vested interest in the success of the system.

Counting on its root, an important goal of PD is to empower users (as a democratic right) so that they will actively take part in decisions; and also enable them develop their skill and knowledge so that they can creatively contribute to the production of a useful and usable solution. The FOSS approach, unlike its closed source counterpart, is more conducive to achieve this objective. In practice, however, power asymmetry is a salient challenge in many FOSS projects, including the successful ones.

PD emphasises the need for software tools to be designed in the context in which they will eventually be deployed [16]. Environments that favour empowerment of users, such as decentralisation of decision-making, open communication, flat organisational structure and progressive leadership are more conducive to apply the PD practices and to exploit its potential [15]. The FOSS literature indicates that such is the organisational nature of many open source projects.

As described in the review section, the FOSS community appears to be technically and theoretically open and invites broad participation, but it is closed for users with limited technical skills. When participating in the development of a piece of software, the peer has to know the programming language, be familiar with the architecture and structure, and capable of participating in the discussions on relevant communication platform.

As the discussions proceed and the project matures, the costs of participation increase. This development has profound implications for the actual openness of a project. That is why we argue that a FOSS project may be legally or technically open, but if it does not provide a set of mechanisms to reduce participation costs, the result is a de facto closed development structure. Moving to an increasingly elitist mode of production will invite only participants who have been following the project long enough to know its intricacies and to have the skills to make a meaningful contribution.

We suggest that PD/DPD can improve the usability and participation problems with in FOSS projects at two levels: during the development of the system, and then during adoption and customization.

During development of the system

The openness of FOSSs enables users to contribute and take part in development, from initiation to final testing and sustainability. Users in traditional design models take part in the design process as informants in the functional analysis phase, or as evaluators in the prototype and simulation phases. However in FOSS development users can be involved in all the phases of the design process [6]. Likewise, PD emphasises the involvement of users at each stage and take place during the whole of the project life cycle [36], which is in line with the organisational nature of FOSS projects.

Participatory Design methods consider users as the domain experts—the ones with the most knowledge about what they do and what they need—and the designers as the technical experts [38]. The fact that users are experts in their field is best utilised if users are able to share their ideas in participative situations. Through participatory design, users are more able to determine their own requirements and map them onto solutions. Developers can then be released from the painstaking task of traditional requirements analysis to concentrate on building systems that will be readily accepted and far easier to maintain [15].

The FOSS development relies on tools and artefacts for cooperation such as mailing lists, discussion forums, and bug-reporting tools [21]. Through an empirical investigation of two open source projects, Iivari [28] indicated that such tools have been successfully used for distributed PD in the FOSS development context in which non-developer users also took part in online discussions concerning understanding, designing and evaluating activities. In so doing, end users play informative, participative or consultative roles.

In FOSS development there is a need to focus on the evolution of software in the presence of a large and active community of users and co-developers [44]. Similarly, an open-ended development perspective is an important feature of PD; it does not attempt to deliver a completed system, instead development is deemed to be open-ended with changing requirements [15].

The underlying reason for employing participatory design is to construct better system designs that take into account the views, requirements, and work of real users [16]. The more pragmatic view of PD also assumes that greater user input and involvement lead to more widespread system acceptance. PD approaches help to meet the of handling software development tasks, as most of the PD techniques focus on issues like vision development, requirements elicitation, and requirements negotiation [26]. These obviously require the direct participation of end users. Proper requirements management is in turn critical to usability issues.

In one of the FOSS cases she studied, Iivari [28] discovered a discussion forum dedicated to usability issues in which different kinds of features are requested, from issues of appearance (how it ought to look) to behaviour (how it ought to behave) and integration (with what it ought to operate). On the one hand, certain appearance or behaviour customisations were also requested. On the other hand, different kinds of problems were expressed; for example, the sender did not know how to use, or does not like, a particular feature of the OSS, or the OSS altogether.

These messages are rampant in many FOSS projects in the form of feature requesting and bug reporting. However, they are useful ways of addressing usability issues from end users as they allow the gathering of information that can potentially improve functionality or usability.

A case study on the Python project [6] highlighted instances of PD practice, which was referred as pushed-by-user proposal. That happens in a distributed manner in the discussion spaces (in the python-list for users and the python-dev mailing-lists) and in the physical interactions spaces, with a specialisation of design topics in these spaces.

The ‘Internet Course Reader’ is another FOSS project which demonstrates the application of PD. It is an educational computer conferencing open source program used for conducting courses via the Internet and it is produced using participatory design principles. To design the program, participants identified their educational computer communications needs. The rest of the project entailed the writing of initial program specifications which were approved by alumni of the residential courses around the world. Meetings with the development team mainly occurred via computer conferencing as well as the creation and presentation of prototypes, follow-up sessions with course participants, field-testing, and the involvement of end users [7].

During adoption and customization

FOSS development obviously poses additional challenges to the application of the traditional PD techniques as it happens in a distributed manner. However, many of the PD techniques can be applied during adoption and customization because a functional system is at hand for end-users to work around from the beginning.

PD reduces the scope and intensity of post implementation training requirements. Selection of competitive software could also be easily done together with the actual end-user themselves, through PD workshops. FOSS systems could be modified and adapted to particular organisational settings together with the PD techniques.

The traditional PD approach has been challenged by physical and organisational distributions. The concept of real participation does not work well with physical distribution because most of the PD techniques are based on the possibility of having face-to-face meetings. PD approaches also fail to address the limitation caused by organisational distribution within the user group or between different user groups.

In contrast to more traditional PD workshops, inter-contextual user workshops [41] related to DPD focus more on reflecting and enhancing use practices within the current system design than on designing new or revised features. This approach, though, is already in practice in many successful FOSS projects.

B. The CBPP model vs. FOSS

In most cases, the FOSS approach is an example of CBPP, but limited to a sub-community of developers, with relatively high entry barriers for average users. We believe that the principles of CBPP can lower these barriers.

First, for mass participation to occur, the modularity and granularity of the artefact is imperative. If relatively large-grained contributions are required, the number of contributors is limited and the process is slowed. If modules are independent, each contributor can independently choose what and when to contribute. This maximises their autonomy and flexibility to define the nature, extent, and timing of their participation in the project.

The CBPP model tells us that the size of the modules (granularity) determines the time and effort that an individual must invest in producing them. The granularity of the modules therefore sets the smallest possible individual investment necessary to participate in a project. If this investment is sufficiently low, then 'incentives' for producing that component of a modular project can be trivial.

A good example of granularity pertains to the Slashdot technology website, where users voluntarily submit and evaluate the issues. Only a few minutes are required for moderating a comment or meta-moderating a moderator. Benkler [10] argues that this is more fine-grained than the hours necessary to participate in writing a bug fix in an open-source project. More people can participate in the former than in the latter, independent of the differences in the knowledge required for participation. His formulation is that the number of people who can participate in a project is inversely related to the size of the smallest scale contribution necessary to produce a usable module.

The cost of integration is also another determining factor; participation requires available and easy to use tools. Another interesting note of the model is that the required time for participation can be drawn from the excess time people normally dedicate to having fun and participating in social interactions. If the grain of contribution is relatively large and would require a large investment of time and effort, the universe of potential contributors decreases.

Raymond [44] noted that for seeing usability bugs, the traditional open source community may comprise the wrong kind of eyeballs. However it may be that by encouraging greater involvement of usability experts and end-users it is the case that given enough user experience reports, all usability issues are shallow. By engaging typical users in the development process, OSS projects can create a networked development community that can do for usability what it has already done for functionality and reliability.

Even though FOSS development strongly relies on Internet tools for communication and cooperation, end users may be unable to fix or report bugs, and communicating with developers through mailing lists and bug-reporting systems might be intimidating for non-developer users [28, 40]. Especially, the reporting of usability-related problems might be complex and difficult to explain textually [4, 28, 56]. Herein lies the advantage of the CBPP approach preaching modular, granular and easy integration mechanisms.

The attributes of the CBPP model thus appear to be suitable in facilitating the intended user engagement in distributed software production. This model, exemplifying the flagship FOSS systems, can also change the balance of throughout the FOSS development approach.

C. *Lessons from CBPP/FOSS to the PD Tradition*

Much like FOSS development, PD/DPD can be helped by modularization by splitting functionality and providing targeting perspectives to specific groups and contexts.

A decade ago, three arenas of participation were being discussed in the participatory systems design tradition: the individual project arena (where specific systems are designed and new organisational forms are created), the company arena (where 'breakdowns' or violations of agreements are diagnosed), and the national arena (where the legal and political framework is negotiated which defines the relations among the industrial partners and sets norms for all work-related issues) [23, 31].

We believe that the peer production approach in general and the FOSS model in particular links the three arenas and even expand the scope to the international level. The hierarchical mode of participation within the three spheres can also be flattened. The decentralised approach helped by the networked environment, therefore, responds to prevailing concerns of losing sight of participation at the latter two arenas [23, 24, 31].

In many PD projects it is not possible for all those affected by the design effort to participate fully [31]. Carefully considering and negotiating the choice of participants with management and workers themselves is a huge task in itself. However, adapting design and development issues to the peer production model could alleviate the problem: interested individuals will come forward, requirements for participation could be fine-grained and the network platform allows off-the-job and/or on-the-job participation at any time. This affects the power relationship, which is a core issue in PD, as it offers equal opportunity

IV. Method

The research approach was a three-year case study [55] of three FOSS projects. The cases were selected on the criteria that they were all FOSS initiatives, but offered quite different solutions. The projects were:

- *Varnish:* An accelerator on web servers for complex web sites or content management systems
- *Skolelinux:* A community-managed FOSS solution aimed at schools (a Custom Debian Distribution)
- *HISP:* A globally distributed open source software initiative, developing health information systems

Data were collected through interviews, online questionnaires and mailing lists. Interviews were conducted with developers, service providers and users, some of them several times. All the interviews conducted were semi-structured, with guiding questions to facilitate the discussions. Online questionnaires were sent to the communities, focusing on motivation, participation and contributions. Among others, the questions pertained to the role of the respondent, the contributions made, the reasons to participate, the challenges faced during participation, availability of mechanisms to attract more users and developers, how the respondent participated in all stages of the development process, and how the technology in use facilitated or constrained contributions.

In addition, the mailing list archives of the three communities were accessed. Documentary sources of the projects were also a useful resource.

Data analysis was conducted as follows. First, each case was analysed chronologically and thematically. Then the perspectives of PD and CBPP were drawn on to enrich the analysis, and to assess alternative or complementary strategies. Then the three cases were compared, and the PD and CBPP perspectives were revisited. This iterative process of sense making [32] was repeated until a consistent analysis was reached. Finally, it was assessed to which degree they were using elements from these approaches, as documented in section 5.

V. Three FOSS Cases

The three approaches discussed in section 3 share useful values and principles. They all represent commitment to an open culture that rejects traditional bureaucratic management styles in favour of a more dynamic and user-centred style with open communications, opportunity for debate and a high degree of personal responsibility and development. This is a necessary foundation for integrating PD principles into FOSS and vice versa, linking them by the key elements of the CBPP production model.

We argue that the three traditions enrich each other. To strengthen our argument we will describe three cases of FOSS development that have used elements from PD and CBPP. Conversely, when these elements are not sufficiently used, we will use this to suggest improvements.

A. *Background to the Three Cases*

Skolelinux : is a community-managed FOSS solution for schools (a Custom Debian Distribution), as an alternative to Microsoft Windows. It was initiated in 2001 by a group of four programmers in Norway who hoped to create a quality, full-fledged and free computer solution for schools. There are now about 25 main developers, with an active core of five. However, more than 170 people to date have directly contributed and committed something on the repository, excluding the larger Debian community. The project in Norway has 15 translators.

General Public License (GPL) is the preferred licensing tool and new contributions are encouraged to comply with it but any license that conforms to the Debian Free Software Guideline (DFSG) is acceptable. DFSG espouses the same ideals as in Open Source Definition (OSI) and Free Software Foundation. The tools in use include SVN for archiving and mailing lists, wiki sites for coordinating development efforts, and a bug reporting tool (Bugzilla).

Skolelinux has branches in Belgium, Brazil, Denmark, Germany, France, Latvia, Norway, Spain and Turkey. It is used in more than 450 schools worldwide -- mostly in Europe but also in Africa. There are between 200 and 30,000 users per school.

Varnish : is hybrid open source software released under the revised BSD (Berkley Software Distribution) license. The project is handled by a company called Linpro in Norway, which mainly produces open source software. Technically, the Varnish software is an HTTP accelerator on web servers for complex web sites or content management systems (CMS).

The project started in early 2006. The software is written entirely in C programming language; Perl is used in some places. All the accompanying tools in the project are open source – Subversion, TRAC, Mailman and GNU author tools.

Linpro makes money by offering add-on services to customers and through sponsorships. The mailing list has names and addresses from all over the world. The main developer works from Denmark on a part-time basis. The other main developer is a full time worker in Linpro, Norway.

HISP : is a globally distributed open source software development which was initiated in South Africa in 1994 and is based on collaboration among academic institutions, health authorities, and private organisations. The goal of the project is enabling south-south and south-south-north collaboration.

The project develops District Health Information System (DHIS), for collecting, processing, and analysing health information for health administration purposes. DHIS 2.0 is a web-based software package released under the BSD license. It is developed using Java frameworks (such as Spring, Hibernate and WebWork) and supported by open source tools mailing lists, Wiki, Subversion (code repository), JIRA (Issue tracker) and IRC channel (for instant messaging).

DHIS 2.0 is developed in globally distributed manner with developers currently in Norway, India, and Vietnam. The software has been implemented in many developing countries in Africa and Asia, such as India, Vietnam, Tajikistan, Sierra Leone, Ethiopia, Tanzania and Zanzibar.

B. Participation Patterns and challenges in the projects

Analysis of the mailing lists of the three projects reveals 82 participants in Varnish, 170 in Skolelinux and 74 in HISP. The development stage and history of the three projects varies. In fact this determines the number of participants that each project could have. In addition, the size of the pool of participants depends on the type and purpose of the software. The messages and threads posted is a good indicator of how active a project is and if participants are taking part in the process. Accordingly, we considered the data of all the projects beginning from their respective starting period. Table 2 summarizes the statistical data about participation and participants in each project.

This data shows the scarcity of participation, and particularly of end-user participation. Skolelinux is said to be used in over 450 schools worldwide. DHIS 2.0 is claimed to be used in more than seven countries, some of them with huge populations. Varnish, by its very nature, is meant to be used by people running large content management systems. This makes the user pool relatively smaller. Its distribution and usage is worldwide, though. According to the table, participation of end-users is still at a minimum. This is indeed a cause for concern in light of the promises offered by the FOSS phenomenon.

An informant from the Varnish project described the difficulty of mastering the learning curve that software development required:

> *As many open source projects, Varnish progress is often slow because of lack of more developers' participation. Perhaps this is also so because of the steep learning curve to be passed before being able to make any significant contribution to the actual Varnish program. It takes a person with high competence in the C programming language and a lot of FreeBSD/Linux kernel knowledge, it seems.*

TABLE 2.

OVERALL PICTURE OF PARTICIPATION IN THE THREE PROJECTS

Project	Year	#ofMessages	#ofThreads	#ofParticipants
Varnish	Feb. 2006 –Dec. 2011	1224	340	82
Skolelinux	Jan. 2002 – Dec. 2011	15548	6344	170
HISP	Nov. 2008 – Dec. 2011	3984	2018	74

In section 2.3, we claimed that projects that strictly follow the CBPP model have fine grained modules which allow participants to easily catch up with the development process. It is argued that such an approach reduces the time and effort required for new contributions.

When asked why the project could not attract more contributors, a core developer of Varnish replied that:

> *It is certainly due to the nature of the market. Varnish only really applies to large web servers so there is no mass-market and there are not as much 'see what I have on my computer' geek-credits in Varnish. Of course the rather advanced*

technological approach taken in Varnish doesn't make it particularly simple for people to jump in, but that is a trade-off we had no choice about since the primary goal of Varnish was to be fast.

This reason serves for many FOSS projects with domain/application specific nature. It argued that most of the iconic FOSS projects that enjoyed success are infrastructural and useful to every person.

A core developer and coordinator of the Varnish project also indicated that no sustained attempt has been made to attract and involve end-users in the development process: “overall, I wouldn't say that we have not made much effort in this area. A press conference was held when release 1.0 came out, but nothing much has happened since.”

One of the interview questions to the developers and/or users involved in the HISP project was to compare the problems associated with FOSS systems to those associated with proprietary products. Here are selected responses from some of the respondents:

‘the installation is difficult’, ‘the user’s interfaces are not very attractive’, ‘there are multiple products and you will be confused on what to choose’, ‘for end-users it is very difficult to use’, ‘you are not sure if the system you get is stable or otherwise’, ‘you don’t get proper support and on time’, ‘the developers are distributed and you don’t get quick help’, ‘documentation is awful so it is hard to learn others code and you got to pay for that’, ‘there is limited awareness in FOSS, usually it is technical gurus who take part’, ‘if you want to customize or modify FOSSs, it requires more technical skill’, ‘ most people don’t know about it. The educational awareness is very low’.

All these issues appear mythically common to the FOSS tradition, but they are yet concerns raised by end users.

C. Features of Participatory Design in the Three Projects

Participatory Design focuses on mutual learning between developers and users, in organised settings of co-operation. The key arena is usually workshops, and two important techniques are brainstorming and prototyping. Table 3 presents the findings in the three projects along the PD features:

A user of Varnish from one of the client companies indicated that he was sending reports in his own way and over time he learnt to write good bug reports. He learned what was needed as he submitted more and more bug reports and from the feedback that he received. By just learning how to interactively work with the developers, he is now continuously assisting them in bug fixing, testing, helping other users, and documentation areas. This case demonstrates the possibility of empowering users through long-term engagement in the development process.

On the other hand, developers can gain domain knowledge and expertise from users, proving that learning in the participation process is a two-way street. One of the main developers in Varnish stated: 'the biggest problem was that I didn't run a big website myself, so I was not intimately aware of all the "standard tricks of the trade", lingo and products people talk about. Their domain knowledge about web servers and related stuffs was very important.'

All three projects have formed communities of interests of their own, including programmers, technical staff, academics and end-users. They operate in a distributed environment, communicating mainly by email and chat channels. Developers communicate on features and solutions, users respond, and report bugs and problems.

Being a core activity in PD, they hold formal conferences and workshops. There is a users and developers conference every year for the Skolelinux project. One of the authors of this paper attended the user conference in Oslo in October, 2008. It was a full-day conference from 10:00 to 17:30. There were around 50 participants, most of them from Norway. There were also representatives and presenters from Germany, Spain, Brazil and Taiwan. The announcement was done through the project’s wiki pages and emails. There were participants from different school municipalities, students, teachers, service providers of the software, developers working full time on the projects, and translators.

The purpose of the conference was experience sharing and introduction of new and future developments. Broader issues were also raised, like how free software meet the needs of the education sector and how such systems are used in schools around the world. There were around seven 45-minute presentations with time allotted for discussions. This shows that the Inter-contextual user workshop introduced by Obendorf et al [41] as Distributed PD is already engrained in the FOSS tradition.

The Varnish project hosts a similar developers and users workshop Varnish project about once a year. The goal is to bring together developers and users to discuss their experiences as well as their requirements.

TABLE 3.

FEATURES OF PD IN THE THREE PROJECTS

	Skolelinux	**Varnish**	**HISP/DHIS 2.0**
Workshops	Formal user conferences	User group meetings twice a year	Training workshops for users and conferences with stakeholders
Brainstorming	Communication through mailing lists and online user forum	Communication over IRC channel and mailing lists	Communication via mailing lists and pilot based face to face meetings
Prototyping	Incremental releases: current version 6.0; ongoing software development	Incremental releases: current version 2.1.5; ongoing software development	Incremental releases: current version 2.8; ongoing software development

The HISP project has a different and yet useful experience in reaching out to end-users. In many places where it enrols the software, it offers trainings for selected representatives and health workers. The first author of this paper had participated in a three-day training workshop in Ethiopia. The training was given for some 30 participants in one of the regional states of the country. The trainees were recruited from remote districts that had installed or that planned to install the DHIS software. The participants were paid per-diem and the workshop was fully sponsored by the HISP project. During the workshop, participants learned the basics of computer use and how to install and use the software. The conference was held in a training centre with a computer for each participant, in a kind of laboratory.

Another useful participatory method observed in the Varnish project is the 'Wish List'. The users submit 'wish lists' to be incorporated in the upcoming versions of the program. This is a kind of one requirements gathering technique employed in the project. A user for example indicated that he has already filed the critical requirements to be incorporated in to the upcoming Version 2.0.

Regarding brainstorming and prototyping, these communities have neither the inclination nor the financial resources to work face-to-face. The question is, then, can electronic communication replace face-to-face co-operation?

As indicated by the HISP [48, 49] and the organising of the Creative Commons licences [51], one solution could be knowledge support networks. In that case, and contrary to most of the literature, participation in a CBPP project may be supported by increasing the skill level of the participating peers through extensive off-line and on-line educational networks. Such networks may be developed within existing educational networks, such the HISP network or the Creative Commons host institution network, or even be the result of corporate sponsoring.

D. *Features of CBPP in the Three Projects*

The hallmark of CBPP is an open architecture that allows members with even modest technical skills to contribute. Thus, the attributes of the IT architecture are very important. Table 4 summarizes how the three cases relate to these aspects.

As illustrated in Table 4, the two first criteria of CBPP, granularity and modularity, are given ample attention in the three projects. Thus, the IT architectures of the cases are well suited to the culture and development of CBPP, by allowing for decentralised development of small contributions.

There is however, one obvious problem. Consider the most famous CBPP solution—Wikipedia--which is the result of thousands of distributed contributors. The coding solution of Wikipedia is characterised by the fact that it takes very little time to learn the codes required to produce new content. However, as is shown here, this is not the case with many FOSS projects. It takes a long time and a great effort for end-users to be able to change the code of Skolelinux, Varnish or HISP.

Yet quality and sustenance are salient challenges, because when barriers are lowered to the minimum, incompetent contribution and errors are inevitable. For example, although Wikipedia's open edit policy and simplicity to contribute are arguably the primary reasons for its success, concerns about quality remain. As Wikipedia continues to mature, it appears to become more difficult to keep up with the proliferation of contributions and edits. This challenge is motivating the development of tools to assist users in creating and maintaining quality [18]. In addition, Wikipedia has instituted a policy that requires contributors to register after making a certain number of anonymous contributions [3].

In fact, apart from easing the participation process for end users, low-cost integration in successful peer production projects includes both quality control over the modules and a mechanism for integrating the contributions (at low cost) into the finished product.
FOSS projects have mechanisms to protect themselves from incompetent or malicious contributions [8, 9] including formal rules, like the GNU General Public License (GPL) that prevents defection, social norms and redundancy of contributions and averaging out of outliers—be they defectors or incompetents. We argue that such mechanisms are more useful than erecting barriers against bad contributions.
The three cases also show that end-users, like students in Skolelinux and health practitioners in HISP, are never active participants in development. It is only when sufficient participation is there that users can exploit their excess capacity in the production process.

TABLE 4.

FEATURES OF CBPP IN THE THREE PROJECTS

Features	Skolelinux	Varnish	HISP/DHIS
Modularity	Collection of individual programs for school purposes; thin client architecture	Some parts are well modularized, easily pluggable and have well designed interfaces; but some parts are not	Individual modules / interfaces on versions 2.0 and above
Granularity	Varies on the individual application	Fairly small subroutines for very specific purposes.	Fairly small modules that can be done independently
Low cost integration	Version controlling using subversion (SVN), easy institutional rules	Subversion (former) and Git for version controlling; consent of the leaders prior to integration is a norm	Subversion (former) and GNU bazaar for version controlling; and guided by the leaders
Sources of excess capacity	Programmers interested in contributing to FOSS in schools; Teachers willing to report bugs.	Programmers hired by Linpro; technical people everywhere use the application running large CMSs	Programmers, researchers, health practitioners who are involved in the HISP network

In Skolelinux, for example, even if the project has more developers now, the end-users' presence in the mailing lists (compared to the claimed number of users, which is in hundreds of thousands) is almost null. This shows the need for a focused effort to ensure substantive user participation, which directly dictates use and usability of systems. It can be argued that partly the nature of the packages, which are collections of many school applications, are suspected of failing to offer fine-grained components for anyone to easily act on them.

E. Summing-up Findings

Concluding this discussion we offer the following findings. First, the FOSS community may improve its usability and its relationship to end-users by adopting some of the principles of PD and CBPP. The most important element from PD is the focus on mutual learning, which is an important extension to the FOSS culture. Regarding the specific techniques of PD, such as brainstorming and prototyping, the FOSS community needs to find ways to do this over electronic media. A number of promising tools can facilitate this.

Regarding the elements from the CBPP approach, as the three cases demonstrate, the IT architecture of FOSS solutions can be designed in a way that allows for local innovation and adaptation, in line with the CBPP approach. FOSS could attract a large number of volunteer end-users to participate – given that modularity, granularity and low cost integration mechanisms are properly handled.

The challenge, therefore, is to create an environment of low-cost integration, allowing non-techies to extend the solution. The costs of participation may be reduced only if the required contribution becomes smaller, hence, we need to increase granularity. Similarly, the integration mechanism has to be affordable both for the collecting entity and the contributing peers.

Another useful observation is that the problems caused by the physical distribution are ameliorated by the global network infrastructure. At least theoretically, the challenges presented by the organisational and temporal distributions can be helped by the features of CBPP. These lessons then can alleviate the challenges of PD and facilitate the transition to DPD.

By engaging typical users in the development process, as Lethondal and Mackay [33] noted, FOSS projects can create a networked development community that can do for usability what it has already done for functionality and reliability. This makes the best use of excess capacities in FOSS projects, one of the core elements underlying the CBPP model.

VI. Limitations and Practical Implication

A. Limitations

As this study bases itself on three cases, it will be difficult to make generalizations on the larger FOSS context. However, the highlighted CBPP and PD issues provide insights for the practitioners and for those who want to conduct similar studies on larger data sets. The heterogeneous nature of FOSS projects in reality also limits the intent of making generalizations.

One such variation lies in the underlying technology that different FOSS projects use. For instance recent version controlling tools could facilitate mass participation in a better way than those that were previous used by the projects studied here. Further studies need to look in to the impact of such emerging technologies, even on the same projects.

The data for the study were mainly obtained from informants who were actively participating in the development and translation. Future research can possibly consult a diverse set of stakeholders, particularly the end users.

B. Implications for Research and Practice

We hope that this study, with its systematic and detailed literature review, complements the IS literature and all distributed software development practices in particular; previous research was limited mainly to commercial project environments.

As the FOSS model of production is spreading to other cultural and knowledge production domains [9, 10], such deeper analysis in to the model is crucial. The attmept made here in addressing its common usability related challanges through revealing linking concepts with PD, can be emperically and theoretically investigated in other domains with similar structure.

The analysis and discussion of the FOSS-related issues contribute to the current ongoing research body on how best to understand the FOSS phenomenon. Equally, practitioners gain from the analysis and discusion as wish to improve participation and usability of their systems.

VII. Conclusion

The free software movement is typified by endowing any user of software with core rights of use, distribution, improvement and study. The FOSS community has produced solutions such as Linux, Firefox and Apache. There are however, reasons for concern for the FOSS production model for end-user software, which is often perceived as being difficult to use.

First, this study reveals that paradoxically, the very model that led to the success of FOSS may fail to produce good end-user software. Several factors explain this finding, the most critical of which is the lack of real end user participation. Because of this, even the assumed and claimed re-balancing of power relationship by the movement does not seem to have fully materialised. It is, therefore, necessary to devise methods enabling wider and deeper participation of end users in design and development.

Second, we suggested that the FOSS community can improve usability and its relationship to end-users by adopting some of the principles and practices of PD, over the distributed network. The most important element from PD is the focus on mutual learning and empowerment, which is an important extension to and from the FOSS culture itself. Regarding the specific techniques of PD, such as brainstorming and prototyping, and also inter-contextual workshop of DPD, the FOSS community needs to strengthen the ways to do this over electronic media. In addition, a more conscious effort to bring new users into the community and encourage their participation is needed.

Third, we argued that the bridge between the FOSS and PD/DPD traditions lies in the implementation and exercise of the CBPP model. This allows the FOSS community to integrate participatory mechanisms for participants without programming skills. This also facilitates the transition of the PD tradition to DPD, given the challenges to this approach.

These changes are easier said than done, but we believe that the emerging social network systems such as Facebook and LinkedIn illustrate the possibility and new ways of approaching this issue. Certainly, if the production model of FOSS is to play a major role in developing end-user software, these are crucial elements. At the same time we should emphasise the deeper lesson: a truly participative process is not one that allows anyone to contribute freely, but the one that provides the infrastructure that users need to acquire the skills to participate in the development process. In this context, then, freedom does equal knowledge. The extent to which this equation has been internalised by most FOSS project still remains to be seen.

References

[1] Andersen-Gott M, Ghinea G, Bygstad B. Why do commercial companies contribute to open source software? *International Journal of Information Management* 12; 32 (2): 106-117.

[2] Andreasen M, Nielsen H, Schrøder S and Stage J. Usability in open source software development: Opinions and Practice. *Information Technology and Control* 2006; 25: 303-312.

[3] Anthony D, Smith S, Williamson T. The quality of open source production: Zealots and Good Samaritans in the case of Wikipedia. Technical Report TR2007-606, Dartmouth College 2007.

[4] Bach P, DeLine R, Carroll J. Designers wanted: participation and the user experience in open source software development. *Proceedings of CHI conference on human factors in computing systems* 2009; 985–994.

[5] Barki H, Hartwick J. Rethinking the Concept of User Involvement, and User Attitude. *MIS Quarterly* 1994; 18(1): 59-79.

[6] Barcellini F, Detienne F, Burkhardt J. User and developer mediation in an open source software community: boundary spanning through cross participation in online discussions. *International Journal of Human-Computer Studies* 2008; 66: 558–570.

[7] Bélanger, M. The Internet CourseReader: An educational computer communications program for organizations in the developing world. Paper presented at the Networked Labour Conference, London School of Economics 2002; 308– 311).

[8] Benkler Y. Coase's Penguin, or Linux and the Nature of the Firm. *Yale Law Journal* 2002; 112(79).

[9] Benkler Y. Sharing Nicely: On Shareable Goods and the Emergence of Sharing as a Modality of Economic Production. *Yale Law Journal* 2004; 114 (85).

[10] Benkler Y. *The Wealth of Networks: How Social Production Transforms Markets and Freedom*. Yale University Press: New Haven, 2006.

[11] Bødker K, Kensing F, Simonsen J. *Participatory IT Design. Designing for Business and Workplace Realities.* MIT press: Cambridge, Massachusetts, 2004.

[12] Boivie I, Gulliksen J, Göransson B. The lonesome cowboy: A study of the usability designer role in systems development. *Interacting with computers* 2006; 18 (3): 601-634.

[13] Bonaccorsi A, Rossi C. Why open source software can succeed. *Research Policy* 2003; 32:1243–1258.

[14] Bygstad B, Ghinea G, Brevik E. Software development methods and usability: Perspectives from a survey in the software industry in Norway. *Interacting with computers* 2008; 20 (3): 375-385.

[15] Cherry C, Macredie R. The Importance of Context in Information System Design: An Assessment of Participatory Design. *Requirements Engineering* 1999; 4: 103-114.

[16] Chin G. A case study in the participatory design of a collaborative science-based learning environment. PhD dissertation, Virginia Polytechnic Institute and State University 2004.

[17] Crowston K, Howison J. Hierarchy and centralization in free and open source software team communications. *Knowledge, Technology and Policy* 2006; 18 (20).

[18] Druck G, Miklau G, McCallum A. Learning to predict the quality of contributions to Wikipedia. *In WIKIAI 08* 2008; 7–12.

[19] Ehn P, Kyng M. *The collective resource approach to systems design*. In Bjerknes, G. et al. (eds.) Computers and Democracy - a Scandinavian Challenge. Avebury, 1987.

[20] Feenberg A. *Questioning Technology*. Routledge: New York, 1999.

[21] Fitzgerald B. The Transformation of Open Source Software. *MIS Quarterly*, 2006; 30: 587–598.

[22] Floyd C, Mehl W, Reisin F, Schmidt G, Wolf G. Out of Scandinavia: Alternative approaches to software design and system development. *Human-Computer Interaction* 1989; 4(4): 253–350.

[23] Gärtner J, Wagner I. Mapping Actors and Agendas: Political Frameworks of Systems Design and Participation. *Human-Computer Interaction* 1996; 11: 187–214.

[24] Greenbaum J, Kyng M. *Design at work: Cooperative design of computer systems*. Hillsdale: NewJersy, 1991.

[25] Grudin J. *Obstacles to Participatory Design in Large Product Development Organizations*. In Schuler D, Namioka A. (Eds.) Participatory design: principles and practices. Erlbaum, 2009.

[26] Gumm D. Distributed Participatory Design: An inherent Paradox? *Proceedings of IRIS29* 2006a.

[27] Gumm D. Distributed Software Development – a Taxonomy. *IEEE Software, Special Issue on GSD* 2006b.

[28] Iivari, N. Participatory design in OSS development: interpretive case studies in company and community OSS development contexts. *Behaviour & Information Technology,* first published in January 2011 (iFirst); 1-15.

[29] Iivari N. Empowering the users? A Critical Textual Analysis of the Role of Users in Open source Software Development. *AI & Soc* 2009; 23: 511-528.

[30] Iivari N. "Constructing the users" in open source software development: An interpretive case study of user participation. *IT & People* 2009; 22: 132-156.

[31] Kensing F, Blomberg J. Participatory Design: Issues and Concerns. *CSCW* 2006; 7:167-185.

[32] Klein H, Myers M. A set of principles for conducting and evaluating interpretive field studies in information systems. *MIS Quarterly* 1999; 23: 67–94.

[33] Letondal C, Mackay W. Participatory programming and the scope of mutual responsibility: balancing scientific, design and software commitment. Proceedings of the eighth conference on Participatory design: Artful integration: interweaving media, materials and practices, 2004.

[34] Loebbecke C, Powell, P. Furthering Distributed Participative Design: Unlocking the Walled Gardens. *Scandinavian Journal of Information Systems* 2009; 21: 77-106.

[35] Luke R, Clement A, Terada R, Bortolussi D, Booth, C, Brooks D, Christ D. The Promise and Perils of a Participatory Approach to Developing an Open Source Community Learning Network. *In Proceedings of the 8th Participatory Design Conference, ACM* 2009; 11-19.

[36] Lynch T, Gregor S. User Participation in Decision Support Systems Development: Influencing System Outcomes. *European Journal of Information Systems* 2004; 13: 286-301.

[37] Matz A. Scattered Locations and Different Stakeholder Views. *Proceedings of the NordiCHI Workshop* 2006.

[38] Namioka H, Rao C. *Introduction to Participatory Design: Field Methods Casebook for Software Design*. John Wiley and Sons: New York, 1996.

[39] Nichols D, Thomson K, Yeates S. Usability and Open-Source Software Development. *ACM, SIGCHI* 2001.

[40] Nichols D, Twidale M. Usability Processes in Open Source Projects. *Softw Process Improv Pract* 2006; 11: 149-162.

[41] Obendorf H, Janneck M, Finck M. Intercontextual Distributed Participatory Design: Communicating Design Philosophy and Enriching User Experience. *Scandinavian Journal of Information Systems* 2009; 21: 53-78.

[42] Pekkola S, Kaarilahti N, Pohjola P. towards Formalised End-User Participation in Information Systems Development Process: Bridging the Gap between Participatory Design and ISD Methodologies. *Proceedings of the 9th conference on PD* 2006; 9: 21-30.

[43] Radtke N, Janssen M, Collofello J. What Makes Free/Libre Open Source Software (FLOSS) Projects Successful? An Agent-Based Model of FLOSS Projects. *International Journal of Open Source Software & Processes* 2009; 1(2): 1-13.

[44] Raymond E. *The Cathedral and the Bazaar*. O'Reilly and Associates: CA, 2001.

[45] Rossi M. Decoding the 'Free/Open Source (F/OSS) Software Puzzle': A Survey of Theoretical and Empirical Contributions. *Quaderni* 2004; 1(424).

[46] Schuler D, Namioka A. *Participatory Design: Principles and practices*. Hillsdale: New Jersey, 1993.

[47] Stallman R. *Free Software, Free Society: Selected Essays of Richard M. Stallman*. In Gay, J. (Eds) GNU Press: Boston, Massachusetts, 2002.

[48] Staring K, Titlestad O. Development as a Free Software: Extending Commons Based Peer Production to the South. International Conference on Information systems. *ICIS* 2008.

[49] Titlestad O, Staring K, Braa J. Distributing Development to Enable User Participation: Multilevel Design in the HISP Network. *Scandinavian Journal of Information Systems* 2009; 21: 27-50.

[50] Torpel B. Participatory Design: A Multi-voiced Effort. In Proceedings of the 4th Decennial Conference on Critical Computing: Between Sense and Sensibility. *ACM* 2005.

[51] Tsiavos P. Cultivating Creative Commons: From Creative Regulation to Regulatory Commons. London School of Economics, London (2007) Available at: http://www.lse.ac.uk/collections/informationSystems/pdf/theses/tsiavos.pdf

[52] Weber S. *The Success of Open Source*. Harvard University Press: Cambridge, Massachusetts and London, 2004.

[53] Ye Y, Kishida, K. Toward an understanding of the motivation Open Source Software developers. *Proceedings of the 25th International Conference on Software Engineering* 2003.

[54] Yeats D. Open-source Software Development and User-Centered Design: A Study of Open Source Practices and Participants. *A Dissertation in Technical Communication and Rhetoric*. Texas, 2006.

[55] Yin R. *Case Study Research.* Thousand Oaks, California, 2003.

[56] Zhao L, Deek, F. Improving open source software usability. *In: N. Romano, ed. Proceedings of 11th Americas conference on information systems* 2005; 923–928.

[57] Zhao L, Elbaum S. Quality Assurance Under the Open Source Development Model. *The Journal of Systems and Software* 2003; 66: 65-75.