

**FLOSSCom - Using the Principles of Informal Learning Environments of FLOSS
Communities to Improve ICT Supported Formal Education**



Phase 1: Analysis of the Informal Learning Environment of FLOSS Communities

Work Package Number: **02**

Work Package Title: **Report on the Learning Environment of FLOSS Communities**

Deliverable Number: **01 (Nr. 18 of the Interims Report at Table 3.3)**

Coordinator: AUTH/MERIT

Authors:

Rüdiger Glott (MERIT)
Andreas Meiszner (SPI)
Sulayman K. Sowe (AUTH)

Contributors:

Thomas Conolly (University of Paisley)
Ashley Healy (University of Paisley)
Rishab Ghosh (MERIT)
Athanasios Karoulis (AUTH)
Hugo Magalhães (SPI)
Ioannis Stamelos (AUTH)
Martin J. Weller (The Open University)
Daune West (University of Paisley)

Due Date: 31-07-2007

Draft Date: 09-07-2007

Availability: General Public as a Work in Progress Document

License: Creative Commons, Attribution + Noncommercial + ShareAlike (by-nc-sa)

Document Id: FLOSSCOM_WP2_PHASE1_REPORT_v070709_1

TABLE OF CONTENTS

LIST OF TABLES	i
LIST OF FIGURES	ii
EXECUTIVE SUMMARY	1
1. INTRODUCTION	4
1.1. Characteristics of FLOSS and its community	4
1.2. Demography	8
2. THE ROLE OF LEARNING AND KNOWLEDGE EXCHANGE WITHIN THE FLOSS COMMUNITY	13
2.1. Roles and responsibilities of FLOSS community members	13
2.2. Motivations of FLOSS community members	16
3. FLOSS AS A LEARNING ENVIRONMENT	24
3.1. Skills learnt in FLOSS	24
3.2. The value of FLOSS skills	29
4. HOW LEARNING IN FLOSS IS ORGANIZED	33
4.1. Interaction and activities	35
4.2. Learning Resources (non-technical)	44
4.3. Knowledge creation in FLOSS communities	48
4.4. Knowledge exchange in FLOSS communities	52
5. TECHNOLOGICAL LEARNING RESOURCES	59
6. CONCLUSION: FLOSS LEARNING PRINCIPLES	67
6.1. Openness and inclusivity	67
6.2. Volunteering and volatility	68
6.3. Using large-scale networks	68
6.4. Content-richness and specialisation	69
REFERENCES	71

LIST OF TABLES

Table 1: Age of new members when joining the FLOSS community	9
Table 2: Age structure of the FLOSS community, 1991 to 2002	10
Table 3: Initial motivational groups and continuing motivational groups within the FLOSS community	21
Table 4: Skills that are learnt a lot within the FLOSS community	26
Table 5: Skills learned in the FLOSS community - by expertise cohorts	29
Table 6: Performance of different age and experience cohorts in the FLOSS community	30
Table 7: Activity fields and activity groups in the FLOSS community	40
Table 8: Providers' views regarding their motives for providing answers to help information seekers on Apache Usenet	57
Table 9: Providers' reasons for reading or scanning questions on Usenet.....	58
Table 10: Tools for knowledge development and sharing.....	60
Table 11: Tools for relationship and trust building.....	61
Table 12: Learning Processes Initiated and Displayed Through Technological Tools .	63
Table 13: Ways to learn in the FLOSS community	64

LIST OF FIGURES

Figure 1: Inflow of new members in FLOSS community	6
Figure 2: FLOSS community members by profession	12
Figure 3: An organizational structure of a typical FLOSS community	14
Figure 4: Role transition in FLOSS communities.....	16
Figure 5: Motivations for joining and staying within the FLOSS community	19
Figure 6: Knowledge about FLOSS before joining the community.....	25
Figure 7: Skills better learnt in FLOSS than in formal computer science courses – community members’ and employers’ view compared	31
Figure 8: Number of regular contacts of FLOSS community members to other members	35
Figure 9: Number of regular contacts in the FLOSS community and project leadership	36
Figure 10: Total number of projects since joining the FLOSS community	36
Figure 11: Activities performed very often by FLOSS community members	38
Figure 12: Knowledge brokers in mailing lists network	42
Figure 13: Beyond the CoP (Community of Practice) Strategy	51

EXECUTIVE SUMMARY

The general concept behind Free/Libre/Open Source Software (henceforth FLOSS) is making the source code of software accessible to anyone who wants to obtain it. Binaries or executables are available via the Internet and can be 'freely' downloaded and used. Prolific licensing agreements such as the General Public License (GPL) define the rights users have over the product. In the literature, many terms are in use to describe the FLOSS phenomenon. Notably, Free Software (FS), a term used by Free Software Foundation (FSF) and Open Source Software (OSS) used by the Open Source Initiative (OSI). In addition, Free Open Source Software (FOSS), Libre Software (LS), and Free/Libre/Open Source Software (FLOSS) are terms frequently used by researchers. In this report the term FLOSS is used to refer to users' freedom to use, modify, distribute, or even sell the software with little obligations as in propriety or closed source software. When a user modifies the software, he/she can either choose to keep changes made private or altruistically return them to the FLOSS community so that everyone can benefit from his derived work.

FLOSS development did not begin with the inception of the Linux operating system, in 1991. Rather, the concept existed since the formation of SHARE – a working group set up to coordinate the programming work of the IBM 701, in 1952 (Sowe, 2007b). In the academic environment, software development and distribution among researchers and departments is not new, and goes back to the earliest days of software in university environments when software was developed to solve practical problems and could be freely shared. However, what is now true is that FLOSS has fundamentally changed the way we develop and distribute software. Enable by the Internet, geographically distributed individuals voluntarily contribute to a project by means of the Bazaar model (Raymond, 1999). Extensive peer collaboration allows project participants to write code, debug, test, and integrate software. Communities in various projects provide support services such as suggestions for products features, act as distributing organs, answer queries, help new members having problems with the software.

FLOSS development not only exemplifies a viable software development approach, but also a model for the creation of self-learning and self-organizing communities (Sowe, et al. 2006c). FLOSS is also a virtual learning context in which both professional software developers and novice users benefit by leveraging their knowledge and information access repertoire. The context enable them to participate at their own convenience and learn (coding or other software related task) at their own pace (Sowe, et al. 2005; page 297).

Furthermore, learners can conclude on the learning scope themselves (*what to do*) and decide on the method of acquiring the knowledge (*how to do it*). Knowledge acquisition is accomplished, for example, by having access to a large code-base, studying online documentation, asking more experienced members for assistance. In essence knowledge is acquired in FLOSS through *learning by doing*, which represents the drill-and-practice approach in normal constructivist environments. However, the FLOSS virtual learning context is not without its downside. Apart from physical isolation and detachment from *face-to-face* interaction commonly associated with virtual learning environments, learning in FLOSS requires access to Internet and moderate computer facilities. In addition, a high degree of computer literacy, reading, writing, and typing is required to participate effectively. The environment is also not conflict free. Flaming, the feeling of being ignored in a dominant discussion, disinterestedness, long delays in receiving responses from communities, the expenditure of searching through discussion archives to see issues previously raised in the communities, access rights to participate in some community activities are all major concerns in the management of the FLOSS learning context (Sowe, et al. 2005).

In recent times, FLOSS is making inroads not only in business and software industries but in colleges and universities as well. There is increased interest in the FLOSS learning environment (Sowe, et al., 2004; Bacon and Dillion, 2006) and in FLOSS projects as *bazaars of learning* (Sowe, et al., 2006). As Faber (2002) noted, FLOSS is both an alternative teaching methodology and an educational model. The main objective of FLOSSCom is using the principles of informal learning environments of FLOSS communities to improve Information and Communication Technology (ICT) supported formal education. Despite the influence and popularity of FLOSS and the benefit inherent in its methodology, educational institutions have been slow to adapt. This can partly due to the fact that the FLOSS environment is fundamentally different from the formal learning environment in most institutions. The focus of this report is collate and report on the learning activities of individuals in various FLOSS communities. The report benefits from the experiences and expositions of various authors in order to provide a synergy and a fresh look into the learning environment of FLOSS communities. FLOSS communities, like other online communities (e.g. Community of Practice (CoPs)) have many interrelated elements that define the dynamics of the community. As such, this report is divided into 5 major sections, each focusing on a specific aspect of the learning environment of FLOSS communities with a 6th section summarizing the main findings.

Section 1: The first part of the report is an introduction to FLOSS. The historical account of FLOSS is presented together with the many meanings of the term. This is followed by the demographic distribution of FLOSS participants and an explanation of what FLOSS communities are.

Section 2: The next chapter examines the role of learning and knowledge exchange within the FLOSS community. It includes a description of the composition and roles of members within communities and what motivates various groups of individuals to participate in FLOSS.

Section 3: After clarifying the role of learning in the FLOSS community the next chapter focuses on the content of learning processes in FLOSS. This chapter examines both, what skills are learnt and how the skills learnt within the FLOSS community are evaluated. The latter aspect is considered from the perspective of FLOSS community members as well as from the perspective of employers.

Section 4: The fourth chapter examines how learning is organized within the FLOSS community. It examines the interaction between community members with regard to learning processes, the learning resources (except for technological resources) that are available and used within FLOSS communities, and the concrete learning processes.

Section 5: This section looks at the technological resources used within FLOSS communities. FLOSS projects are almost exclusively administered online and one of the most important prerequisites for their coordination and cooperation is provided by the functionality of various communication and groupware tools.

Section 6: The last section of the report proposes the project's approach to learning in FLOSS. A preliminary set of guidelines as they apply to the principles of informal learning environments of FLOSS communities to improve ICT supported formal education are offered. These preliminary guidelines will be continuously amended in the following phase of the FLOSSCom project.

1. INTRODUCTION

1.1. Characteristics of FLOSS and its community

Software can be allocated to two very different domains. The first one is determined by so-called proprietary software. Proprietary software is distributed on the basis of licenses that keep the property of the software to the software company that produced it. The user has to accept the software as it is sold by the producer and its vendors and to pay license fees for using it. The buyer and user of a proprietary software product is thus not allowed and not capable to read and change the source code of the program.¹ If the user wants to customize functionalities of a proprietary software system to his needs he relies on the producer's willingness to take his demand into account when the next release of the software program is created. The world market for desktop operating systems, for instance, is dominated by proprietary software, as 84% of all desktop operating systems are Microsoft Windows XP.²

The other software domain is Free/Libre and Open Source Software (FLOSS)³. In contrast to proprietary software, FLOSS is distributed on the basis of licenses that entitle the users to read and modify the source code and that do not require the user to pay license fees for using FLOSS systems. Well-known examples of FLOSS are the Linux operating system or the Apache web server.⁴

What constitutes FLOSS – apart from the license under which it is distributed - is the Internet-based collaborative development and sharing of non-proprietary software. FLOSS developers

¹ Vendors of proprietary software provide the users only with object or binary code, which can be read by the machine, but not by users.

² March 2007, according to Net Applications. See <http://marketshare.hitslink.com/report.aspx?qprid=2>,

³ In the literature, many terms are used to describe software whose source code is accessible to anyone who wants to obtain it. The term 'free software' was popularised in the mid of the 1980s by the Free Software Foundation, which was founded by Richard Stallman (see www.fsf.org). The term 'open source software' was established in 1997/98 by the Open Source Initiative (see www.opensource.org). The term "libre" refers to the respective term in Romanic languages, such as "logiciel libre" in French, and at the same time it is used to underline the political meaning of the term "free" in "free software", as it does not imply that software is distributed for free, but that it is distributed in order to provide consumers and the society more freedom to use and develop software (see Stallman, 2002). In this document, the term FLOSS is used to refer to users' freedom to use, modify, distribute, or even sell the software with little obligations as in proprietary or closed source software (Sowe, et.al.2007).

⁴ Apache is web server software used on "web server" computers connected to the Internet and Usenet is its field support system, taking the form of publicly accessible "newsgroup" discussion forums. In 2005, the Apache web server held a share of about 70% on the world market for web server operating systems, as indicated by NETCRAFT LTD, May 2005 Web survey of 63,532,742 hosts, available online at http://news.netcraft.com/archives/2005/05/01/may_2005_web_server_survey.html. For a detailed discussion of FLOSS market shares see the FLOSSIMPACT report (Ghosh et al. 2006 - online at: <http://ec.europa.eu/enterprise/ict/policy/doc/2006-11-20-flossimpact.pdf>)

send the code they developed to repositories for FLOSS code and applications, such as Sourceforge (<http://sourceforge.net/>) or Freshmeat (<http://freshmeat.net/>). From these repositories, code and applications can be downloaded, tested, and – if necessary or possible – improved and finally resubmitted by other users. These repositories do not only provide access to the software source code, they also provide means for communication, such as forums or mailing lists (both are also widely used in big FLOSS projects).⁵

The ideal conception of a FLOSS community is that of an egalitarian network of programmers developing software in a decentralized environment free of hierarchical control structures. However, the popularity of FLOSS coupled with the success of some products (eg Linux base Operating Systems such as Ubuntu, Debian, or Redhat) has changed the ecology and dynamics of FLOSS communities. Large numbers of technical and non-technical end-users are now participating in FLOSS projects [Lakhani and Hippel, 2003; Michlmayr, 2004]. These individuals are not necessarily hackers⁶ but they get involved in activities that are essential for the FLOSS development process (Fitzgerald, 2004), as well as the maintenance and diffusion of the software (Michlmayr, 2004; Sowe, *et.al*, 2006).

When a user modifies the software, he/she can either choose to keep changes made private or return them to the FLOSS community so that everyone can benefit from his [derived] work. In the latter case, any improvements on the current version of the software are also made freely available. The FLOSS development process exemplifies a viable software development approach. The key for this opportunity is the plurality of FLOSS licenses, of which the best-known is probably the General Public License (or GPL).⁷ Each license defines exactly the rights the user has over the product. FLOSS participants are usually geographically

⁵ Just to provide an impression of the dimensions of such repositories: At the time of writing this paper, 144,999 projects and 1,553,410 users were registered at Sourceforge, which claims to be the largest FLOSS repositories. It should be noted in this context that only very active members of the FLOSS community are registered at such repositories and that the community is much larger than indicated by these figures. We will show in later sections of this report that contributing to FLOSS is not limited to writing code but includes a broad variety of activities that do not require any affiliation with a FLOSS repository.

⁶ According to wikipedia.org: A hacker is a software designer and programmer who builds programs and systems that garner the respect of one's peers. A hacker can also be a programmer who hacks or reaches a goal by employing a series of modifications to exploit or extend existing code or resources. For some, "hacker" has a negative connotation and refers to a person who "hacks" or uses kludges to accomplish programming tasks that are ugly, inelegant, and inefficient. This pejorative form of the noun "hack" is even used among users of the positive sense of "hacker. Outside the FLOSS community the term "hackers" is often assigned to people who illegally intrude IT systems of others. In the FLOSS community, these persons are named "crackers". See, for instance, Raymond's "New Hacker Dictionary" (also called "the jargon file") at http://www.ccil.org/jargon/jargon_toc.html

⁷ For an overview see, for instance, <http://www.gnu.org/licenses/license-list.html>

distributed (sometimes all over the world), and a good part of their collaboration is through the Internet, using project’s mailing lists, emails, discussion forums, Internet Relay Chats, etc. Software development activities are usually coordinated through *de facto* versioning systems such as Concurrent Versions System (CVS) or Subversion (SVN), bug-tracking systems (BTS) and bug databases (e.g. Bugzilla), mailing lists, forums, Internet Relay Chats (IRC), etc. These tools not only enable participants to collaborate in the software development process but also act as repositories to store the communication activities of the participants. The FLOSS-model has produced a number of successful applications in the area of operating systems (Linux), emailing and web services (Sendmail, Apache), databases (MySQL, PostgreSQL), and recently also conquers the office desktop (OpenOffice.org), to mention a few.

FLOSS is not a new or young phenomenon. As revealed by the FLOSS Developers Survey (Ghosh et al. 2002), the main dynamics of FLOSS development took place in the second half of the nineties (just after the market success of Linux). Some community members claim to have started with FLOSS already in the fifties (see figure 1).

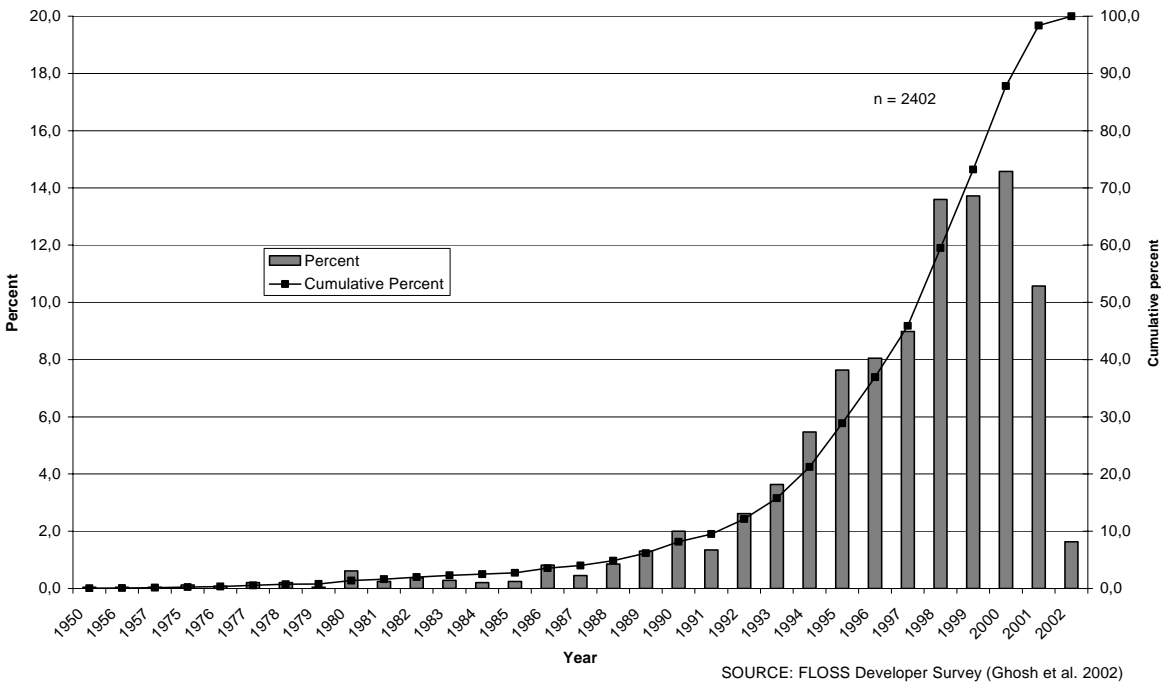


Figure 1: Inflow of new members in FLOSS community

FLOSS communities are special kinds of online communities where the focus is on software development, maintenance, and support. The Internet, particularly the Web, continues to support the emergence and proliferation of [virtual] communities of diverse interest (Sowe, *et.*

al., 2004). Virtual communities are attracting a lot of attention as valuable communities through which both learners and instructors can try new approaches. The pedestal on which they are built continues to influence teaching and learning, and content quality and delivery. Howard Rheingold (1993) defined virtual communities as "social aggregations that emerge from the net when enough people carry on those public discussions long enough, with sufficient human feeling, to form webs of personal relationships in cyberspace". Thus, there is social interaction taking place in cyberspace, resulting in the establishment of relationships among the participating personas. Dowdy (2001) sees this as an intimate secondary relationship: an informal, frequent, and supportive type of community relationship that operates only in one specialized sphere of influence, such as F/OSS. As a good source for acquiring and sharing information (Rheingold, 1993; Lapachet, 1994), virtual community members may be involved in practical information seeking (seeking answers to specific questions in the form of postings). In some communities, members undertake a more general activity characterized by frequent visits to monitor the information neighborhood and what goes on in various communities (Burnett, 2000). Erdelez (1999) likens this activity to information encountering: a memorable experience of an unexpected discovery of useful or interesting information. To a certain degree, the FLOSS community can be considered as a large community of practice (or as a reservoir of many such communities of practice), since membership in this community is not defined by official status but by participation, it develops informally around things that matter to people (in this case: software), and it produces a shared repertoire of communal resources (routines, sensibilities, artefacts, vocabulary, styles, etc.) (Wenger 2000; Ghosh et al. 2005a) .

As agents of socialization and information providers, virtual communities and communities of practice serve as important learning environments. Sowe et al. (2004, 2005) therefore concluded that FLOSS is also a model for the creation of self-learning and self-organizing communities. FLOSS communities can therefore be described as web-based learning communities (Sowe, *et. al.*, 2004) in which individuals interact with collaborating peers to solve a particular problem and exchange ideas. Collaborative learning and the peer review process emphasize the importance of shared dialogue. In this regard, the principles and practices of learning in the FLOSS community appear helpful to master the challenges coming up with the growing demand for "lifelong learning" (OECD, 1977; Livingstone, 1999). These challenges consist mainly in the necessity of new leaning arrangements that are more informal, self-organized, and incidental (i.e. driven rather by situational personal interests and needs than by pre-defined curriculae of educational institutions or firms) (Keeton

et al., 1976; Houle, 1976; Chickering, 1976; Coleman, 1976, 1995; Lave & Wenger, 1991; Watkins & Marsick, 1992; Cseh et al., 2000; Council of Europe, 2000; Dohmen, 2001; Overwien, 1999). David & Foray (2002) describe this change as an overall shift from “learning to do” to “learning to learn”. Given the potentials provided by the Internet, “communities of practice” (Brown & Duguid 1991), especially Internet communities are considered to be extremely successful in developing and deploying such new learning forms. FLOSS communities often serve as prime examples when the capacities of such volatile network organizations are demonstrated (Faust & Holm, 2001; Demil & Lecocq, 2003; Hemetsberger. & Reinhardt, 2004; von Hippel, 2002; David & Foray, 2002; von Krogh et al., 2003). In fact, the FLOSS developer survey (Ghosh et al. 2002) has revealed that this community is mainly driven by its members’ individual wish to learn about and share knowledge of the development of open source software, its philosophy, and the cooperation within the community (Ghosh et al., 2002; Ghosh et al., 2004). Thus, the open source community provides a suitable object for research on the mechanisms and structures that characterise the new forms of learning (Ghosh & Glott 2006) .

1.2. Demography

The FLOSS Developers Survey (Ghosh et al. 2002) revealed that the age of the FLOSS community members ranges from 14 to 73 years, while there is a clear predominance of people between 16 and 36 years. Only 25 per cent of the respondents are older than 30 years, and only 10 per cent are older than 35. The average age (mean) of the respondents is 27.1 years.⁸ Despite the astonishingly high maximum age we can thus conclude that members of the FLOSS community are extraordinarily young, usually around the mid of the twenties.

The starting age of the FLOSS community members ranges between 10 and 55 years, whereby only 7 per cent started below an age of 16 years, one third was between 16 and 20 years old, another third between 21 and 25, and a quarter was older than 26 years when starting developing FLOSS. The average starting age is 22.9.⁹

Ghosh et al. (2005a: 158) explained these observations as follows: “Altogether, the results suggest that developing FLOSS is more populated by the younger generation than by

⁸ The standard deviation is 7.2 years, which indicates quite a large spread within such a young group.

⁹ Standard deviation: 6.32 years.

experienced software developers. However, taking into account that FLOSS is by no means a phenomenon of only the recent years, the young age of developers cannot be explained only by generational effects. Changes in the frame settings for the production of software – like increasing capital investments in FLOSS projects and the new organization structure that is performed in FLOSS projects (Lerner & Tirole 2000, p. 1) together with the growth of the Internet seem to be the key factors for the increasing attractiveness of FLOSS for young people.”

Table 1 and table 2 illustrate the attractiveness of the FLOSS community for young people further by relating the age when joining the community (table 1) and the current age (table 2) to the year of joining the community.

Age when joining the FLOSS community	Period / Year joining the FLOSS community										
	1950 - 1985	1986 - 1990	1991 - 1995	1996	1997	1998	1999	2000	2001	2002	Total
10 - 15 years	16,1	12,2	10,2	5,7	4,6	5,6	5,2	6,6	0,8		6,6
16 - 18 years	27,4	17,6	15,7	24,2	22,0	20,1	13,6	16,5	10,1	8,1	17,1
19 - 21 years	19,4	25,2	24,9	22,2	32,6	26,2	27,3	19,1	28,0	16,2	25,1
22 - 25 years	11,3	24,4	25,7	26,3	22,5	26,9	25,8	30,5	28,8	32,4	26,3
Total: 10-25 years	74,2	79,4	76,5	78,4	81,7	78,7	71,8	72,6	67,7	56,8	75,1
26 - 30 years	21,0	12,2	12,4	13,4	14,2	12,3	17,6	18,2	17,9	27,0	15,2
Older than 30 years	4,8	8,4	11,0	8,2	4,1	9,0	10,6	9,1	14,4	16,2	9,7
Total: 26 years and older	25,8	20,6	23,5	21,6	18,3	21,3	28,2	27,4	32,3	43,2	24,9
Total	100	100	100	100	100	100	100	100	100	100	100

n = 2402

p < 0.01; Contingency coefficient: .225

Source: FLOSS Survey (Ghosh et al. 2002)

Table 1: Age of new members when joining the FLOSS community

The figures in table 1 seem to imply that to some degree the shares of juvenile community members decline over time, but it must be taken into account that the FLOSS community in 1990 represented only 8% of the size of the community that was reached in 2002, as indicated by the cumulative percent curve in figure 1.

In fact, if we consider the actual age of the community members at the time the FLOSS survey was conducted and only for the period of 1991-2002¹⁰, we find that the share of the youngest age group in the community is continuously increasing between 1991 and 2000 (table 2).

Current age (spring 2002)	Period / Year when joining the FLOSS community								
	1991 - 1995	1996	1997	1998	1999	2000	2001	2002	Total
14-23 years	13,1	29,9	39,4	43,8	46,1	52,1	52,5	40,5	35,0
24-30 years	50,2	48,5	45,9	42,3	37,9	35,3	33,1	43,2	40,2
older than 30 years	36,7	21,6	14,7	13,9	16,1	12,5	14,4	16,2	24,9
Total	100	100	100	100	100	100	100	100	100

n = 2402

$p < 0.01$; Contingency coefficient: .447

Source: FLOSS Survey (Ghosh et al. 2002)

Table 2: Age structure of the FLOSS community, 1991 to 2002

Women are under-represented in FLOSS; only 1.1 per cent of the respondents to the FLOSS Developers Survey (Ghosh et al. 2002) were women. The FLOSSPOLs Survey on Gender Issues (Krieger et al. 2006) therefore directly approached women's organization within the FLOSS community but could also find only a 5%-share of female FLOSS community members. Although women are clearly underrepresented in programmers in industrialized countries, too, this extremely low share of women in the FLOSS community cannot be compared to gender structures in IT professions: According to Suriya (2003), the share of women in programmers in the end of the 1990s and beginning new millennium varied between 22% (North and Latin America, UK, Germany) and 33% (Finland). The FLOSSPOLs Gender Study revealed a number of reasons for this, which altogether result in the conclusion that women are actively excluded by the male dominated FLOSS community.

This active exclusion is based on "a central cultural dynamic within FLOSS. FLOSS participants, as in most scientific cultures, view technology as an autonomous field, separate from people. This means that anything they interpret as 'social' is easily dismissed as 'artificial' social conditioning. Because this 'conditioning' is considered more or less arbitrary, in their view it is supposed to be easily cast aside by individuals choosing to ignore it. FLOSS also has a deeply voluntarist ethos which values notions of individual autonomy

¹⁰ It is self-evident that the share of young community members declines the more the period under consideration lies in the past, but focussing on the last ten years before the survey was conducted limits this bias to a justifiable degree.

and volition. As a result, participants largely do not believe that gender has anything to do with their own individual actions. The situation is thereby perpetuated in spite of the expressed desire for change” (Krieger et al. 2006, page 144).

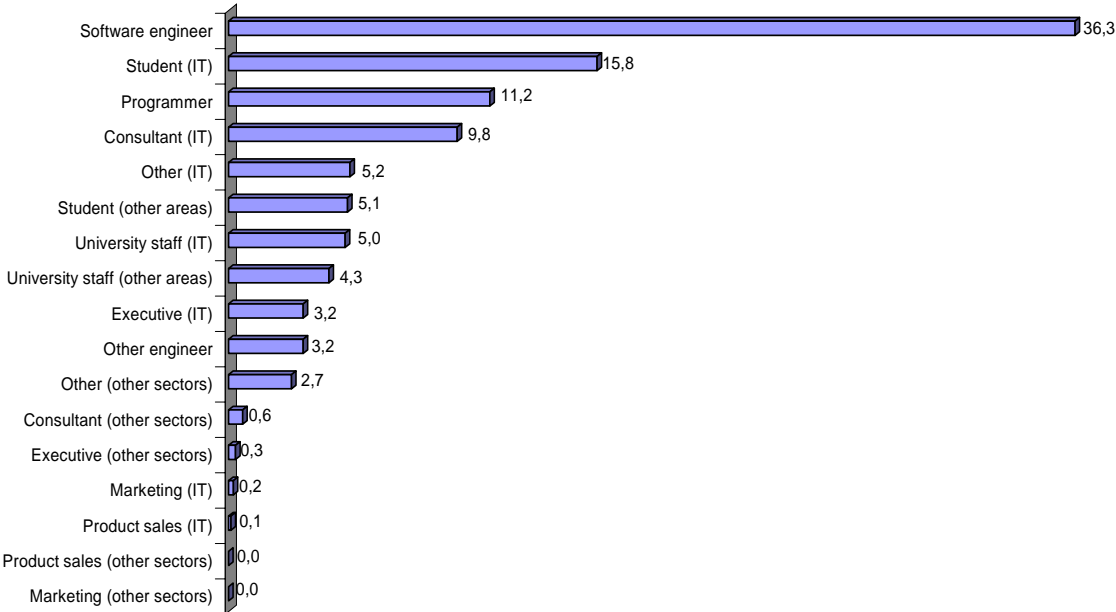
One means of active exclusion of women (which may happen unconsciously) is mainly exercised through communication styles, because inflammatory talk and aggressive posturing (‘flaming’) is accepted within many FLOSS projects as a key means of developing reputation. “This is often off-putting for newcomers and less experienced contributors who are not yet familiar with the community, its norms, or its real hierarchy. The effect is particularly pronounced in the case of women, who in most cases have a shorter history in computing and therefore less confidence in defending themselves on technical grounds. ‘Flaming’ thus exacerbates the confidence difficulties women tend to have as a result of lower levels of previous computing experiences.” (Krieger et al. 2006: 145)

Another factor that contributes to active exclusion of women is the “hacker ethic” that FLOSS communities actively perpetuate, “which situates itself outside the ‘mainstream’ sociality, but equates women with that mainstream. Women are treated as either alien or (in online contexts) are assumed to be male and thus made invisible. Women are seen as innately more able to organise, communicate and negotiate among F/LOSS projects as well as with the outside world. Thereby they become carriers of sociality that is seen in a contrast to the ‘technical’ realm ascribed to men. Additionally F/LOSS women receive a high level of attention due to their gender which decreases their feeling of acceptance as community members as well as their willingness to further engage with the community” (Krieger et al. 2006, page 145).

FLOSS community members reward the producing code rather than the producing software and thereby put most emphasis on a particular skill set while other activities such as interface design or documentation are understood as less ‘technical’ and therefore less prestigious. The latter set of skills is often associated with women, which again puts them in a disadvantaged position within the FLOSS community.

FLOSS community members are typically people with a higher education, as 9% have a PhD and 70% have a university degree. Another 17 per cent of FLOSS developers have a high school degree (the rest is still in education or has a lower educational degree) (Ghosh et al, 2005). This observation is explained by the fact that programming requires capacities in abstract and formal reasoning, which is usually developed in the course of higher educational attainment.

Figure 2 shows the professional structure of the FLOSS community members, as revealed in the FLOSS Developers Survey (Ghosh et al. 2002, 2005). Professions and university courses related to the IT sector dominate, as 83% have a profession related to this sector or deal with similar tasks at universities. Software engineers provide one third of the sample and thus the largest single group. With a share of 16 per cent, students are the second largest group, followed by programmers and IT consultants. Executives, marketing and product sales experts do not have a significant impact on the professional structure of the FLOSS community.



Source: FLOSS Developer Survey 2002

Figure 2: FLOSS community members by profession

The share of those who are unemployed or not working for other reasons is 4.8%, pure students provide 14.5 % of the FLOSS community, semi-professionals (students who have an IT-related job) have a share of 8.4 %, and the large majority of the community (73.3%) is made up by employed professionals, whereby the share of self-employed is 14%. (Ghosh et al. 2005: 160)

2. THE ROLE OF LEARNING AND KNOWLEDGE EXCHANGE WITHIN THE FLOSS COMMUNITY

In order to evaluate the role of learning and knowledge exchange we first have to get a better understanding of how the FLOSS community is structured, beyond pure demographics. What is relevant here is which roles different (groups of) community members play within the community and which responsibilities they take for development projects, interaction, coordination, socio-political movement et cetera. Once these roles and responsibilities are understood we will turn to what motivates the FLOSS community members to contribute to FLOSS. This question is usually what puzzles people outside the FLOSS community very much because it is hard to understand why the community members invest so much time and effort into something they often are not paid for. As we will see, individual demand for learning and knowledge creation and the FLOSS community's capacity to meet this demand is the key to answering this question.

2.1. Roles and responsibilities of FLOSS community members

There are different types of memberships and roles to be found in FLOSS communities. At the very heart of each community are the project initiators and the core development team. A review of the top 100 FLOSS communities of sourceforge.net (Krishnamurthy, 2002) concludes that the core development team consists of a very small number of people, meanwhile the bigger part of the core team, the enhanced team, is involved in other tasks like e.g. providing feature suggestions, trying products out as lead users, answering questions etc. (Gosh, 2005; Krishnamurthy, 2002). Crowston and Howison (Crowston, 2004) suggested a hierarchical or onion-like structure for FLOSS communities (Figure 3a), consisting of the following layers:

- At the center of the community are the core developers, who contribute most of the code and oversee the design and evolution of the project.
- In the next ring out are the co-developers who submit patches (e.g. bug fixes) which are reviewed and checked in by core developers.
- Further out are the active users who do not contribute code but provide use-cases and bug-reports as well as testing new releases.
- Further out still, and with a virtually unknowable boundary, are the passive users of the software who do not speak on the project's lists or forums.



Figure 3: An organizational structure of a typical FLOSS community

The onion model is the most referenced model of a sustainable FLOSS community. The term "Onion", according to Aberdour (2007) refers to the "successive layers of member types" and can be described thus;

"Individuals create the sustainable community by increasing their involvement through a process of role meritocracy. As they move toward the core, users become bug reporters and might over time become contributing developers. A few contributing developers will eventually join the small team of core developers. Each type of member has certain responsibilities in the system's evolution, which relate to the system's overall quality. Advancement through the member types is reward and recognition for each member's abilities and achievements", (Aberdour, 2007; page 59)

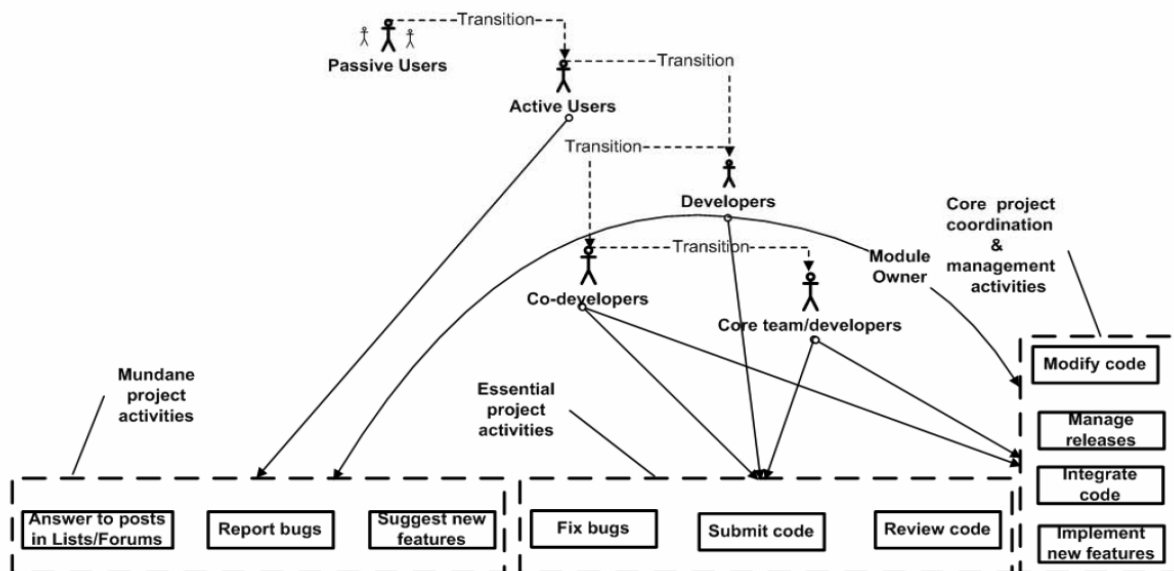
In their study of role migration and advancement processes in the Mozilla, Apache, and NetBeans projects, Jensen and Scacchi (2007) criticized the onion model view of FLOSS communities. They argued that in its present form the onion model fails to draw out the presence of multiple tracks of project career advancement through different role-sets. They suggested an onion-pyramid view of the organizational structure of FLOSS communities as shown in Figure 3b.

The enhanced team is engaged in other multiple forms of leadership like thought leaders, networkers, people who document the practice, pioneers, etc. (Wenger, 2000). But, the largest part of the community is the users group that rather "consumes" than to "contribute". This largest part is classified by Giuri *et al.* (2004) as the "external contributor"; meanwhile the remaining parts are seen to be the "internal member".

To just provide an idea about the possible distribution of high active core member, less active member, and the wider active community it is referred the Apache case study that reported a distribution of 15 core members contributing 83% of the code, 250 regular member in different functions and 3000 active member of the wider community that mainly report bugs (Giuri 2004). Besides the contribution of code or content (programs, artifacts, execution

scripts, code reviews, comments, etc.) members also actively engage in online discussion forums or threaded email messages as a regular way to both observe and contribute to discussions of topics of interest to community participants (Scacchi, 2001). It could further be observed that community members choose on occasion to author and publish technical reports or scholarly research papers about their software development efforts, which are publicly available for subsequent examination and review. Each of these highlighted items point to the public availability of data that can be collected, analyzed, and represented (Scacchi, 2001).

FLOSS communities are very flexible communities where roles are not stagnant. Rather than having the stringent structures present in traditional software development organizations, most advancement or promotion in FLOSS communities is through meritocracy. A handful of people have earned the right to make decisions based on merit or because of their past contributions. In some projects (e.g. Debian, FreeBSD, Apache) positions are filled through a democratic voting process. The success of a project is best measured by the success of its community. A great community is capable of producing great code, a community with quality participants will create quality code. More importantly, a great community is one which opens its doors to others who are perceived to be less hacker-like. A great community is capable of lowering barriers for potential developers and novice users. Of course a good community leadership is important in creating a good community around a project. A good community or project leader is someone who listens to the voices from within the community, forge relationships and give credits where it is due. Let the community feel that they created and own the software. Describing community structures in terms of concentric circles does not limit members advancing from one layer or role to another. Individuals create a sustainable community by increasing their involvement through a process of role meritocracy (Sowe, 2007; Jensen and Scacchi, 2007), which varies from one project to another. Through sustained contribution, users are recognized and gradually move from one role to another. Passive users may become active users and might, in due course, become developers or co-developers. A few contributing co-developers will eventually join the small team of core developers. These role transitions are depicted in figure 4.



Source: Sowe 2007: 51

Figure 4: Role transition in FLOSS communities

The dotted boxes shown in figure 4 show some of the project activities members in each role may be involved in. Active project participants are mostly involved in mundane project activities. Essential project activities such as reviewing, approving, and committing code to the project's source tree are mostly done by the trusted core and co-developers. Developers who may have earned sufficient credits with the core team may also contribute to essential project activities. In projects like Debian where some developers are maintainers of packages, they take complete responsibility of their packages by coordinating and managing activities associated with the package.

2.2. Motivations of FLOSS community members

In the course of the success of FLOSS programs like Linux and the Apache Web server the question why people spend efforts and time gratis for FLOSS increasingly triggered research on the FLOSS phenomenon. According to Lerner and Tirole (2002), all contributions to FLOSS, though free of charge, can be explained by economic motivations because behind all these contributions they assume the wish of the contributors to signal their skills and experience to the wider community and to thus enhance their chances on the labour market. This view is however based on theoretical considerations of why people might contribute freely to FLOSS and narrows the whole phenomenon down on reputation issues. It does not

include other values of the FLOSS community are also important for its understanding and that contradict the idea of a primary economic motivation as driver of all contributions to FLOSS, as they are expressed in Raymond's (1999) "bazaar" model or by the Free Software Foundation (www.gnu.org/philosophy). Such values are, for instance, a deep interest in and need for software and the conviction that software should be freely available and provide the users the opportunity to check and modify the code. As Demaziere et al. (2006: 8) point out: "We have mostly met computer programmers for whom the commitment to free software had neutral, even negative consequences, from a material point of view. Above all the validity of the hypothesis of motivation through financial incentives is founded on the premise of a contribution based on a calculated choice, anticipating the long term effects on a career. Yet, what our interviews show is that it is a more progressive commitment, sustained by a growing familiarity with programming activity and the "social world" of developers (Strauss, 1978) and accentuated by memorable experiences through which computer programmers build a sense of participation and interaction with other free software developers."

Referring to Raymond's (1999) "bazaar" model and focusing on basic tasks that must be performed in FLOSS projects, Lakhani and von Hippel (2002: 923) identify following "major motives used to explain why users would voluntarily work on these basic tasks:

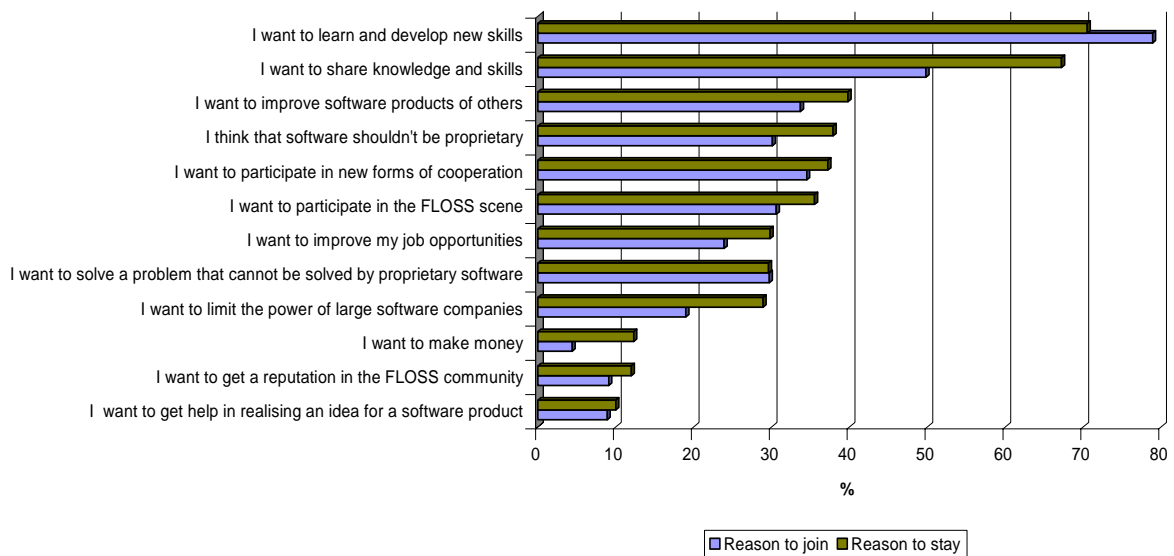
- a user's direct need for the software and software improvements worked upon;
- enjoyment of the work itself; and
- the enhanced reputation that may flow from making high-quality contributions to an open source project."

However, Lakhani and von Hippel (2002) point out that these three motivations do not suffice to explain the motivation to perform "mundane but necessary" tasks in FLOSS projects, such as providing "field support", which is provision of help to people having problems with a FLOSS product.

Kollock (1999), as referred to in Lakhani and von Hippel (2002: 927) also points out that reputation enhancement may drive people to contribute to FLOSS, but he also stresses other factors that may motivate FLOSS community members: expectations of reciprocity that can reward providing something of value to another; the act of contributing that can have a positive effect on contributors' sense of efficacy (i.e. a sense that they have some effect on the environment); and contributors' attachment or commitment to a particular open source project or group.

Lakhani and von Hippel (2002) developed an approach towards motivations of FLOSS community members that goes beyond theoretical considerations and anecdotal evidence by directly addressing the activities carried out by community members and asking them for their reasons to carry them out. Their approach is however limited in the sense that it focuses only on one major activity, the provision of help to others, and one FLOSS project (the Apache web server). The range of activities carried out within the FLOSS community is however very broad - whereby providing help to others is not among the most important of these activities – and that many of these activities are not bound to participating in a specific project. The FLOSS Developer Survey (Ghosh et al., 2002) therefore aimed at examining motivations related to the FLOSS community as a whole.¹¹ The survey also acknowledged that members of the FLOSS community have different degrees of knowledge and expertise in FLOSS-related issues and that therefore knowledge creation and sharing might also play a role as motivators for contributing to the community. Finally, the FLOSS Developer Survey targeted the motives to participate in the FLOSS community from two different perspectives, the reasons for joining the community and the reasons for staying in the community. Figure 5 illustrates the answers to these two questions, whereby the respondents have been asked to limit themselves to a maximum of four answers. The answers are ordered by the size of the shares allocated to the “reasons to stay in the community”, since these reasons help explaining how the community manages to become sustainable.

¹¹ Overall, 2783 FLOSS community members (mainly European and North American, but also from South America and Asia) participated in this survey.



Source: FLOSS Developer Survey 2002

Figure 5: Motivations for joining and staying within the FLOSS community

Evidently, improving skills and sharing knowledge are the most important motivators for people to engage in FLOSS. The second important group of motivators is provided software-related aspects (wish to improve software products), political aspects (software should not be a proprietary product) and by aspects of socialising within the community (participation, collaboration). Except for “learning and developing new skills”, which motivating power declines by nature with growing expertise over time, all these reasons gain importance after the community member got some experience, which particularly applies to the product-related and the political item.

Motives such as “to solve a problem that could not be solved by proprietary software” or “to get help in realising a good idea for a software product”, but also a material motive (“to improve my job opportunities”) reach shares between 20% and 30% of respondents. While the motive to get help in realizing an idea for a software product shows no change in its importance, the other two items, especially the motive to improve job opportunities by contributing to FLOSS, gain considerably importance.

The wish to limit the power of large software companies has been a motivating factor to join

the community for approximately one fifth of the community and shows a very strong increase with growing experience within the community.

“To make money” can be neglected as a motivator for joining the FLOSS community (it plays mainly a role for consultants at the age of above 30 years (i.e. with probably some years of professional experience in their occupation). However, it should be noted that this factor gains some importance once community members have become more familiar with FLOSS.

To get a reputation and to help realising ideas for software products are the two least important motivators for joining as well as for staying in the community.

As a conclusion, what motivates people to join FLOSS cannot be explained by “simple economics” (Lerner & Tirole 2002), which becomes evident in particular by the low shares of respondents who say that getting a reputation within the FLOSS community and making money are motivators. Rather than that, the provision of a learning environment in which new skills can be developed and knowledge is shared motivates the largest part of the community to contribute to FLOSS.

A factor analysis with a subsequent cluster analysis identified six diverging groups of developers with regard to the initial motivations (that are reasons to join the FLOSS community)¹² and four diverging groups with regard to continuing motivations (that are reasons to stay in the community). Table 3 shows these motivational groups and how initial motivational groups and continuing motivational groups are related to each other.

¹² It must be noted that these initial motivational groups are not built by only those FLOSS community members that belong to the community for a short period of time. We asked all community members to remember their initial motivations and to tell us about their current motivation to stay within the community. Therefore, all respondents to these two question belong to a certain initial motivational group *and* at the same time to a certain continuing motivational group. The following sections in the text how these groups relate to each other.

		INITIAL MOTIVATIONAL GROUPS					
		Enthusiasts 4% of respondents	Software improvers 10% of respondents	Recognition seekers 11% of respondents	Materialists 12% of respondents	Ideologists 17% of respondents	Undecided 46% of respondents
CONTINUING MOTIVATIONAL GROUPS	Recognition seekers 12% of respondent	33,7	6,0	41,6	14,1	7,2	5,8
	Skills improvers 33% of respondents	32,7	22,8	37,0	26,3	43,6	31,1
	Software improvers 24% of respondents	30,8	37,5	13,0	39,4	14,4	23,2
	Ideologists 31% of respondents	2,9	33,7	8,4	20,2	34,7	39,9

Source: FLOSS Developer Survey 2002, Ghosh et al. 2005: 171)

Table 3: Initial motivational groups and continuing motivational groups within the FLOSS community

The first initial motivational group is enthusiastic about FLOSS in many respects, except for monetary and career concerns. Its members show an extremely strong impact of a desire for software and skills improvements, but also of ideological aspects and reputation. The “software improvers” shows only a strong impact of software related motivations, while skills-related motivations and monetary and career concerns do not motivate this group. The third initial motivational group is provided by “recognition seekers”, i.e. FLOSS community members strongly driven by a wish for getting a reputation in the community. While monetary and career concerns still play a role as motivators in this group, ideological motivations play no role at all, and the role of software related motivations is also limited. “Materialists”, with a strong and exclusive motivation by money and career concerns, provide the fourth initial motivational group. The fifth initial motivational group within the FLOSS community consists of “ideologists” who are clearly driven by the wish to abolish proprietary software in favour of free software and who want to limit the power of large software companies. The “undecided” appear opposed to the “enthusiasts”. While enthusiasts are driven by almost all facets of FLOSS, the undecided show quite disparate motivating patterns with regard to their reason to join the FLOSS community.

Apparently, most people joining the community have no clear concept of what they expect from this step and what its outcome will be. Ideological motives are the dominant driving

force among those who have an idea of their purpose to join the FLOSS community.

Regarding groups that emerge when reasons for staying within the community are considered, we observe dissolution of the initial enthusiasts, materialists, and undecided to the benefit of a new group that is strongly driven by the wish for skills improvement. Recognition seekers and software improvers remain, which may reflect processes of professionalism and objectification. Ideologists either do not dissolve. Moreover, as shown in Table 3, people who continue participating in the FLOSS community for ideological reasons provide a considerable proportion of the FLOSS community, equally large as the proportion of the skills improvers. While software improvers still provide roughly one quarter of the FLOSS community, recognition seekers play the least important role in quantitative respect. It appears also noteworthy that recognition seekers hardly attract those who have been undecided when joining the community. Together with the fact that materialists dissolved this illustrates that learning skills and promoting the philosophy of FLOSS drives people more than economic benefits that may derive from their contributions to the community.

The demographics of the motivational groups show significant differences: For the initial motivational groups we observe that

- enthusiasts are typically married, older than 30 years, and have comparably low educational degrees; they are usually employed, but provide also the highest share of unemployed compared to the other initial motivational groups; they are typically software engineers and programmers belonging to the FLOSS community for quite a while (typically more than 5 years)
- software improvers show similar demographic structures, but have on average higher educational degrees and their professional backbone in consultants, managers, and university staff
- recognition seekers are strongly determined by students, though featuring also above average shares of software engineers and programmers; they are typically single and comparably young, belonging to the community for a medium or short period of time
- materialists are typically living in partnerships; like enthusiasts and software improvers they are typically older than 30 years, but in contrast to these two groups they are obviously starters in the FLOSS community, as indicated by the above average share in those who belong to the community for one year or less; they usually provide a bachelor degree and are employed or self-employed, typically as software

engineers, but also as consultants or managers

- ideologists are, like recognition seekers, determined by students, but in contrast to this group they show a comparably high share of people over 30 years and – correspondingly - a higher educational degree and of employed; they typically belong to the community for a short period of time
- the undecided, due to the fact that they provide the lion share of the initial motivational groups, do not differ notably from the average

Regarding the continuing motivational groups, we found that

- recognition seekers are typically singles, very young (and therefore with a high share of students), and provide comparably low educational degrees; they provide a relatively high share of unemployed and typically belong to the community for a medium period of time
- skills improvers resemble very the recognition seekers but show less unemployed and a shorter period of time of belonging to the community
- continuing software improvers differ in some respects from their counterpart among the initial motivational groups, since they are typically married, medium-aged, and self-employed, while their typical professions and their period of membership in the community do not differ from initial software improvers
- ideologists are comparably old-aged, have high educational degrees, and show a relatively high share of people who do not belong to the community for more than one year

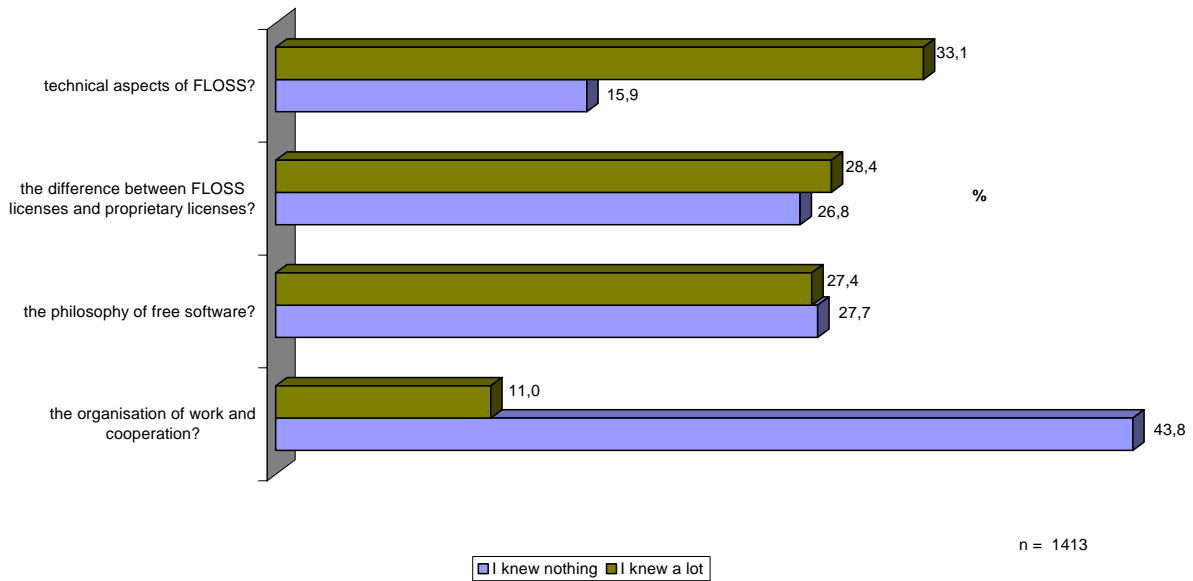
3. FLOSS AS A LEARNING ENVIRONMENT

The FLOSS development process exemplifies a viable software development approach. It is also a model for the creation of self-learning (Sowe et al., 2004) and self-organizing communities (Valverde, 2006) in which geographically distributed individuals contribute to build a particular application. The model has produced a number of successful applications in the area of operating systems (Linux), emailing and web services (Gmail, Apache), databases (MySQL, PostgreSQL), etc. The success of these products has changed the ecology and dynamics of FLOSS communities. Large numbers of technical and non-technical end-users are participating in FLOSS projects (Fitzgerald, 2004; Nikolas, *et al.*, 2003). They get involved in activities that are essential for the FLOSS development process (Fitzgerald, 2004), as well as the maintenance and diffusion of the software (Michlmayr, 2004; Lakhani and Hippel, 2003). These studies provide great insight into the collaborative software development process that characterizes FLOSS projects and users' involvement. The aforementioned studies provide insight into the FLOSS development process as well as community involvement in FLOSS projects. As noted by Staring in 2005 "the study of open source approaches represents a rich and largely untapped resource for institutionalized education" (Staring, 2005). Although rapidly growing the current number of studies on FLOSS communities from an educational point of view is still relatively limited. This was also confirmed by Bacon and Dillon's position paper on the potential of FLOSS approaches in education, by highlighting that there are still no empirical studies available which offer a reliable insight into the potential uses of FLOSS approaches in education.

3.1. Skills learnt in FLOSS

The FLOSSPOLLS Developer Survey (Ghosh & Glott 2005) reveals that a considerable share of the FLOSS community members knew already quite a lot about FLOSS at the time they join the community (Figure 6). One third knew already a lot about technical aspects of FLOSS, and more than one fourth, respectively, knew a lot about differences between software licenses and the philosophy of FLOSS. Only the way how work and collaboration is organized in the FLOSS community appears as quite unknown to newcomers.

At the time you joined the FLOSS community, how much did you already know about ...



Source: FLOSSPOLS Developer Survey 2005

Figure 6: Knowledge about FLOSS before joining the community

45% - 50% of the respondents say that they knew “a little” about all four aspects of FLOSS we have asked for in the FLOSSPOLS Developer Survey (not displayed in Figure 6). As a conclusion, for a good part of the newcomers learning about FLOSS within the community does not start by zero, they have already basic knowledge about most of the fundamental aspects of FLOSS before they join the community.

To integrate and/or play a role in the FLOSS community is not per se and not only dependent on good programming skills, it can also require expertise in patents law and license issues or management skills and capacities to mobilise community members in order to exercise pressure on political and economic decision-makers. Since the lingua franca of the community is English and since many software projects ask for translations of the code and programme documentation into other languages (so-called “localisation”), language skills are also required in the FLOSS community. All these skills can however be learnt within the community, through interaction with other community members and project participation. These skills are however not equally important for all individual members. Table 4 illustrates that coding skills are the ones that are considered to be learned best within the FLOSS community, whereas the shares of those saying they learned a lot of managerial skills appear lowest in comparison to the other skills. This does however not imply that managerial skills

are not learnt – the community members focus on coding skills because software development provides the centre of the CoPs in FLOSS. As we will see later, managerial skills play quite an important role when skills learnt in the FLOSS community are compared to formal courses.

Skills learned in FLOSS	I learned a lot
To understand the differences between copyrights, patents, and licences	51,7
To re-use code written by others	48,0
Basic / introductory programming skills	47,7
To run and maintain complex software systems	47,1
To understand licences	47,0
To write code in a way that it can be re-used	46,0
To understand copyright law issues	44,7
To become familiar with different programming languages	44,3
To get an overview of developments in software technology	44,3
To understand patent law issues	40,0
To look for and fix bugs	38,7
To better understand English, especially technical discussion	37,0
To design modular code	35,5
To get an overview of the skills you need in the software professions	33,2
To clearly articulate an argument	27,3
To express personal opinions	27,2
To accept and to respond to criticism from others	27,0
To improve your understanding of liability issues	23,5
To coordinate own work with the work of others	23,4
To understand and work with people from different cultures	22,9
To document code	22,4
To interact with other people	18,6
To evaluate the work of others	18,5
To lead a project or a group of people	17,4
To keep a community going	15,8
To clearly define and achieve targets	14,3
To motivate people	13,0
To create new algorithms	12,4
To settle conflicts within a group	12,4
To plan work and stick to a work schedule	7,1

n = 1453

(Multiple responses ordered by shares (%) of respondents) Source: FLOSSPOLs Developer Survey 2005

Table 4: Skills that are learnt a lot within the FLOSS community

Before we examine these ways of knowledge transfer in detail in the following sections we will show that indeed the community is extremely successful in managing this transfer. For this purpose and in order to examine how different degrees of expertise affect the learning

behaviour within the community we distinguish 12 different types of FLOSS community members by their experience, which we measure by age and sojourn time within the community:

Young novices (15-26 years old, up to 3 years in community):	7.9 %
Young semi-experienced (15-26 years, 4-5 years):	10.7 %
Young experienced (15-26 years, 6-7 years):	9.7 %
Young experts (15-26 years, >7 years):	6.9 %
Middle-aged novices (27-32 years, up to 3 years):	3.1 %
Middle-aged semi-experienced (27-32 years, 4-5 years):	5.6 %
Middle-aged experienced (27-32 years, 6-7 years):	8.1 %
Middle-aged experts (27-32 years, >7 years):	16.8 %
Old novices (>32 years, up to 3 years):	2.1 %
Old semi-experienced (>32 years, 4-5 years):	3.8 %
Old experienced (>32 years, 6-7 years):	5.6 %
Old experts (>32 years, >7 years):	19.8 %

We use the combination of these two features because an indicator of expertise that is based on project-experience alone would fall short of the complexity of experience that can be made in the FLOSS community and because age provides a hint for different socio-demographic, especially professional backgrounds of community members. It can be assumed that a person joining the community at the age of 15-20 has no profession or just started to learn a profession, whereas a person joining the community at the age of 26 or older probably has already some professional experience.

Table 5 shows the shares of the 12 expertise cohorts who answered that they learned some or a lot of these skills. Above average shares are highlighted and skills that showed no differences between cohorts are excluded.

The young cohort of the community shows the strongest skills improvement, in form of a kind

of “learning curve” that is characterised by a continuous expansion of skills improvements across and within all fields over the four expertise groups. The “learning curve” starts with the social skill “to express personal opinions” and the general skills to improve English capacities and to interact with other people. The semi-experienced young community members show an increasing scope of skills improvements in these two fields, but the main characteristic of this cohort is that programming skills are improved a lot. The experienced young community members show an increase of the scope of skills improvements that is observed within the semi-experienced, but they exceed this scope by skills improvements in managerial and legal skills, too. Finally, the young experts show typical improvements of all skills except for two legal skills and the technical skill “to run and maintain complex software systems”.

In contrast to the “learning curve” of the young cohort, the middle-aged cohort is characterised by alternating expansion and contraction of the scope of skills that are improved over the four expertise groups, with changing focus on different skill fields. The novices focus on improving basic / introductory programming skills, which they obviously try to achieve by looking for bugs and fixing them. But they also typically try to motivate others, which implies that they have a concrete objective aligned with their entry into the community. The middle-aged semi-experienced report a much broader set of skills they see improved through participating in the FLOSS community, mainly managerial, legal, and social skills. The experienced middle-aged community members show a decreased scope of skills that are improved as compared to the semi-experienced, and in contrast to their counterparts they focus on technical and legal skills. Finally, the middle-aged experts observe skills improvements on an equally broad scope than the semi-experienced, but they focus on technical and managerial skills.

The old cohort within the FLOSS community shows no typical profile of skills improvement among the novices and semi-experienced. The skills improvement profile of the old experienced community members is characterised by a focus on legal and social skills, whereas the old experts focus very much on managerial skills.

Overall, these observations illustrate that learning behaviour differs considerably between different groups of community members with regard to the degree of experience the group members have. The most striking finding is that apparently young community members – who lack professional experience that the older cohorts make outside the community - find in the FLOSS community a learning environment that allows them going through a completely informal but very comprehensive curriculum in order to become full-fledged FLOSS

programmers, coordinators and activists.

		Expertise cohorts (age, sojourn time in community in years)												
		young novices (15-26, 0-3)	young semi-experienced (15-26, 4-5)	young experienced (15-26, 6-7)	young experts (15-26, >7)	middle-aged novices (27-32, 0-3)	middle-aged semi-experienced (27-32, 4-5)	middle-aged experienced (27-32, 6-7)	middle-aged experts (27-32, >7)	old novices (33-66, 0-3)	old semi-experienced (33-66, 4-5)	old experienced (33-66, 6-7)	old experts (33-66, >7)	Total
		n=111	n=150	n=136	n=97	n=44	n=79	n=114	n=237	n=29	n=53	n=79	n=278	n=1407
Technical skills	Basic / introductory programming skills	73.1	83.7	80.3	78.9	80.0	72.2	66.1	75.8	53.6	45.7	63.2	63.8	71.7
	To write code in a way that it can be re-used	74.3	84.2	83.0	84.0	75.0	71.1	73.9	85.3	69.2	55.3	69.3	75.7	77.8
	To re-use code written by others	77.4	83.9	84.3	90.4	75.6	81.1	89.1	86.1	76.9	77.1	75.0	83.6	83.3
	To document code	50.0	62.8	71.4	66.0	56.1	56.8	63.6	59.5	33.3	52.2	46.5	62.3	59.3
	To become familiar with different programming languages	67.0	76.4	80.7	89.4	65.0	75.3	81.7	79.7	66.7	72.9	71.1	70.3	75.8
	To design modular code	67.3	69.2	80.3	78.7	70.7	62.2	68.8	77.4	55.6	41.3	50.7	68.2	69.4
	To run and maintain complex software systems	70.9	81.6	81.7	78.3	63.4	75.9	75.5	85.3	55.6	67.4	71.6	76.0	77.1
	To look for and fix bugs	60.2	76.4	72.0	76.6	75.0	69.7	77.8	80.4	50.0	48.9	64.5	72.6	72.2
	To clearly define and achieve targets	42.6	41.3	50.8	59.3	41.7	58.7	39.8	47.7	42.9	35.7	43.5	50.2	47.3
	To evaluate the work of others	58.5	52.5	69.0	69.8	54.1	54.7	62.1	63.5	53.6	42.9	55.2	62.4	60.4
Managerial skills	To coordinate own work with the work of others	48.4	54.9	65.1	72.1	51.4	66.2	56.4	66.0	44.4	33.3	59.1	66.8	60.9
	To lead a project or a group of people	32.6	42.6	45.6	48.2	41.7	50.0	42.3	35.7	27.3	47.7	51.4	43.7	
Social skills	To express personal opinions	57.9	66.7	74.6	73.0	51.3	74.0	60.8	66.7	46.4	43.2	64.3	56.7	63.4
	To clearly articulate an argument	54.8	68.1	74.0	70.5	59.0	73.6	64.4	64.4	46.4	51.1	55.1	59.4	63.5
	To accept and to respond to criticism from others	58.9	72.2	74.6	77.5	59.0	67.1	62.7	65.3	39.3	50.0	73.9	66.1	66.6
	To settle conflicts within a group	30.4	35.0	49.2	52.3	41.7	47.9	45.9	42.3	21.4	19.5	50.7	49.2	43.2
	To motivate people	33.0	37.9	39.8	61.9	45.7	49.3	37.4	38.4	39.3	32.6	48.5	46.6	42.4
Legal skills	To understand copyright law issues	68.0	75.7	78.0	81.7	78.9	88.2	81.7	80.8	64.3	70.0	78.4	79.6	78.4
	To understand patent law issues	67.3	70.9	73.3	81.7	73.0	82.7	78.9	78.9	60.7	61.2	80.0	76.0	75.3
	To understand licences	81.6	83.9	85.6	82.6	78.9	89.5	91.8	86.8	71.4	74.5	82.7	85.5	85.0
	To understand the differences between copyrights, patents, and licences	78.8	76.5	87.1	89.0	73.7	85.3	89.0	84.1	78.6	68.0	85.1	85.1	83.2
	To improve understanding of liability issues	42.3	57.5	62.4	65.2	51.4	68.5	59.8	58.3	46.4	47.8	62.2	63.9	59.1
General skills	To better understand English, especially technical discussion	72.5	72.8	74.8	65.6	42.9	65.8	67.0	63.6	32.1	33.3	50.0	47.1	60.8
	To interact with other people	64.0	68.2	64.9	71.7	60.5	64.5	65.4	58.1	42.9	45.5	47.2	53.1	60.0
	To understand and work with people from different cultures	56.1	65.1	68.2	68.9	48.7	63.2	48.1	63.8	53.6	38.6	63.0	61.2	60.8
	To get an overview of the skills you need in the software professions	72.0	75.5	80.0	76.7	69.2	71.6	75.5	79.6	57.1	60.9	73.0	69.8	73.9

Source: FLOSSPOLs Developers Survey

Table 5: Skills learned in the FLOSS community - by expertise cohorts

3.2. The value of FLOSS skills

As Hemetsberger & Reinhardt (2006a; 2006b) point out, the FLOSS community must find ways to transfer the knowledge necessary to integrate and perform in the community to newcomers. Like table 5, table 6 implies that people join the community at different age, with different backgrounds, different capacities and resources, and with different objectives. The middle-aged novices have already the knowledge and skills to start immediately working on projects while the young and the old apparently involve themselves in projects after they stayed in the community for a significant period of time. Obviously, the middle-aged attain these skills somewhere outside the community before they become active in the community. The young obviously do not dispose of such external experience, they have to learn in all project-related issues, but the longer they stay in the community the more active and experienced they become. Young experts provide the highest degrees of project involvement, project leadership, and networking. Finally, the old community members show comparably low degrees of project involvement and project leadership, but they show significant networking activities. These differences may not only relate to different interests and capacities of these groups. They are probably also partially due to technological changes and the changed role of the FLOSS community for software markets and the information society.

Compared to the pioneers of the FLOSS community, which are mainly represented by the “experts” of the old and the middle-aged group, novices, semi-experienced and experienced community members found a relatively large number of existing projects and a well developed infrastructure for communicating, dealing with code, and learning.

Expertise cohorts (age, sojourn time in FLOSS community in years)	In how many FLOSS projects are you involved at present?	In how many FLOSS projects did you participate since you joined the community?	And how many FLOSS projects did you lead since you joined the community?	With how many Free / Libre / Open Source Software developers do you have regular contact?
young novices (15-25, 0-3)	2,0	8,7	0,8	10,3
young semi-experienced (15-25, 4-5)	2,5	6,6	1,5	8,3
young experienced (15-25, 6-7)	3,0	9,4	1,2	12,6
young experts (15-25, >7)	3,9	15,9	2,2	17,1
middle-aged novices (26-32, 0-3)	3,3	16,9	0,7	5,6
middle-aged semi-experienced (26-32, 4-5)	2,7	4,8	1,3	9,5
middle-aged experienced (26-32, 6-7)	3,3	8,1	1,5	11,0
middle-aged experts (26-32, >7)	3,4	9,5	1,8	12,8
old novices (33-66, 0-3)	2,0	3,2	0,6	13,6
old semi-experienced (33-66, 4-5)	2,4	6,0	0,6	8,3
old experienced (33-66, 6-7)	2,3	6,2	1,1	15,6
old experts (33-66, >7)	3,4	10,1	1,7	11,8
Total (average)	3,0	9,1	1,4	11,7

Numbers in cells are mean values

Source: FLOSSPOLS Developers Survey

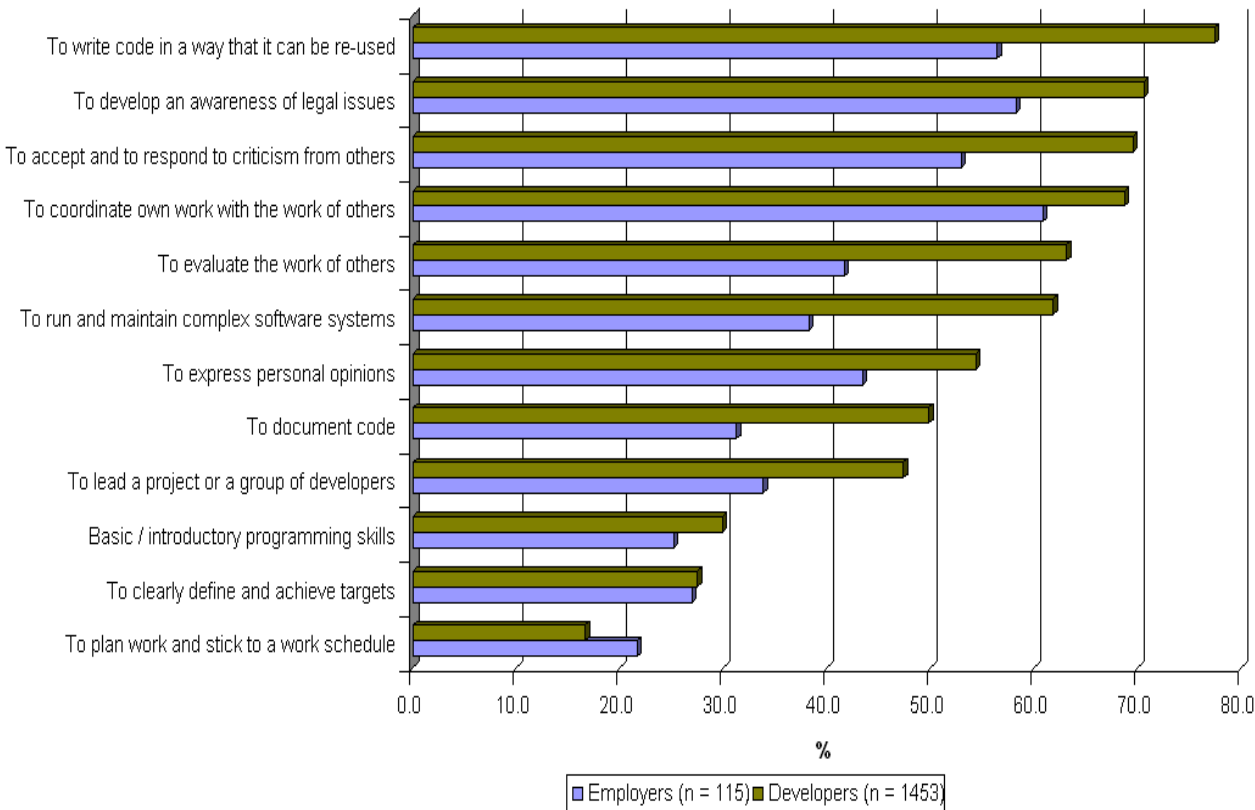
Table 6: Performance of different age and experience cohorts in the FLOSS community

With regard to the core interest of the FLOSSCOM project two questions regarding the learning processes in the FLOSS community appear crucial: The first is whether or not FLOSS can provide skills better than formal computer science courses; the second is whether skills learnt in FLOSS can compete with formal degrees on the labour market.

Figure 7 shows how developers and employers¹³ assess the skills that can be learned in the FLOSS community as compared to formal computer science courses. Evidently FLOSS skills are favoured stronger by the community members than by employers, which is explained by the fact that FLOSS community members identify stronger with their community than human resources managers of companies.

¹³ Data on employers is taken from the FLOSSPOLS Employer Survey that was carried out by MERIT in 2004. Human resources managers of 115 companies that are experienced in FLOSS have participated in this survey. A reservation must be made with regard to the validity of the results of the employer survey, as 71% of the respondents say FLOSS plays an important or very important role for their company. .

To write re-usable code, to get an awareness of legal issues, to deal with criticism, and to coordinate own work with the work of others are the five skills that get most assent from community members as skills that can better be learned in FLOSS than in formal computer science courses. Interestingly, the first four of these skills are also the ones that get most assent from employers, though the internal order of these four differs from the one of the community members. Overall, these findings illustrate that in the perspective of community members as well as in the perspective of employers the FLOSS community provides a broad variety of skills –technical (coding), managerial, and legal skills –better than formal computer science courses.



Sources: FLOSSPOLs Developer Survey 2005, FLOSSPOLs Employer Survey 2005

Figure 7: Skills better learnt in FLOSS than in formal computer science courses – community members’ and employers’ view compared

Community members as well as employers regard informal skills attained in FLOSS as very competitive on the labour market. 85% of the employers say that FLOSS experience adds value to formal computer science degrees. 38% of the employers consider informal qualifications and formal qualifications to be equal, 29% think that formal qualifications are better, and 17% consider informal qualifications to be better. 16% do not know which of the

two qualifications is better. Four fifths of the FLOSS community members are convinced that proven FLOSS experience can compensate for a lack of formal degrees, and three fifths consider the skills they learn within the FLOSS community as core skills for their professional career.

4. HOW LEARNING IN FLOSS IS ORGANIZED

FLOSS production is a highly collaborative construction process targeted at solving complex problems (Scharff 2002), which was coined by Benkler (Benkler 2002) as commons-based peer production (CBPP). CBPP "refers to any coordinated, (chiefly) internet-based effort whereby volunteers contribute project components, and there exists some process to combine them to produce a unified intellectual work. CBPP covers many different types of intellectual output, from software to libraries of quantitative data to human-readable documents (manuals, books, encyclopedias, reviews, blogs, periodicals, and more)." as summarized by Krowne (Krowne 2005). Scharff (Scharff 2002) therefore described FLOSS communities as collaborative learning environments and analysed the principles of how open source communities collaboratively solve complex problems and identified the following characteristics as being relevant:

- Open source participants engage in personally meaningful activities.

"In open source, projects must be designed so that a potentially large group of individuals with diverse motivations can all carve out personally meaningful chunks. Open source communities engage in collaborative discussion and construction to learn their mutual needs and negotiate to produce something meaningful for everyone. In particular, communities use collaborative technology to share ideas (and software) and often engage in conscious creation of priorities in the form of "wish lists" and "TODO items"."

- Designing software helps create solutions to ill-structured problems.

"In complex design activities, there are often no optimal solutions or any straightforward notions of the "right answer." Learning in these contexts is not about the acquisition of facts but involves the continuous activities of framing and solving problems."

- Communities rely heavily on shared external representations.

"Externalizations are important because they are easily shared and reflected upon."

- Collaborative Technologies are used extensively.

"In most open source projects, computer-based collaborative tools are the primary media for communication."

- Contributions are incremental and continuously integrated.

"Open source communities provide examples of how boundaries in learning activities can be more fluid. Many interesting problems are continuous, even though we lack the ability to

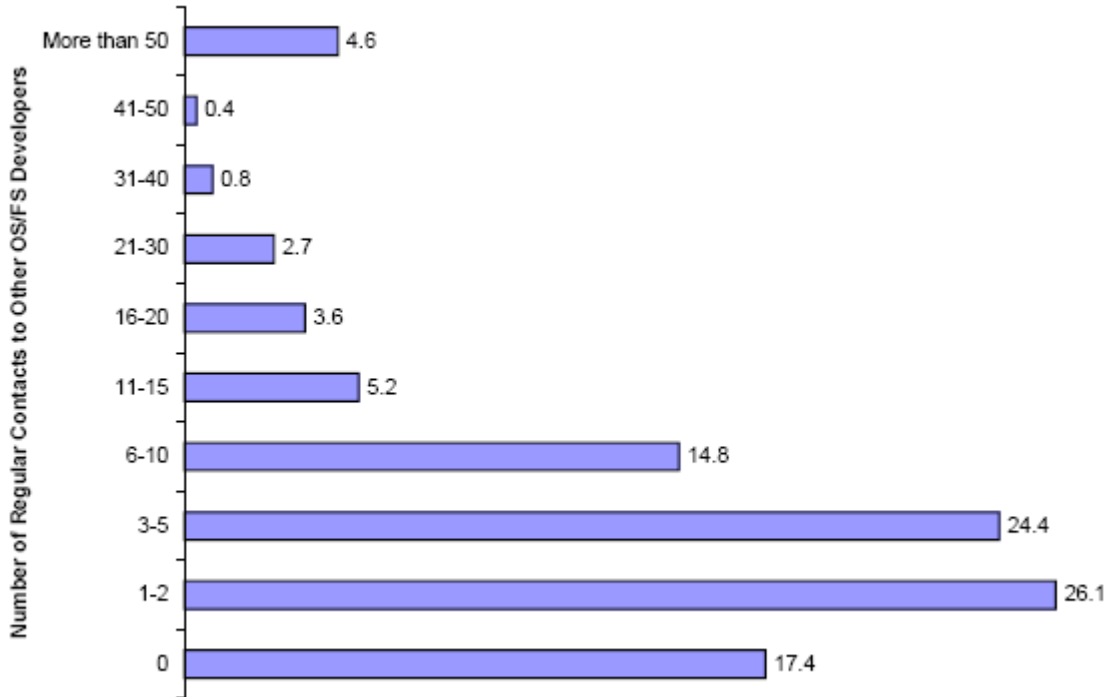
handle ongoing activities. Learning situations are often constrained by numbers of participants, available time, geographic locality, and so on. Open source communities are not static.”¹⁴

Learning is always to a great deal an individual exercise it always depends on external factors. Collaborative learning in the FLOSS community is highly dependent on the interaction between community members and the resources the community provides. These aspects are examined in the following sections of this chapter.

¹⁴ Though Scharff might not have been aware about, with the last point (Contributions are incremental and continuously integrated) he actually describes a process of continuous improvement (CPI), also known as Kaizen. The concept is frequently used within the business sector. The philosophy behind is that the management understands that revealed mistakes in the past are a chance to improvement in the future. Therefore the goal of CPI is to use the potential of the employees to improve all processes on a continuous base to maximize the performance of the organization. Along with this CPI is seen to increase the quality of processes, products and services, as well as the production output. In FLOSS this is certainly the case, and understanding FLOSS as learning environments one could talk about a learning environment that is in a continuous improvement situation where new learner don't start at scratch but build upon past experiences of other.

4.1. Interaction and activities

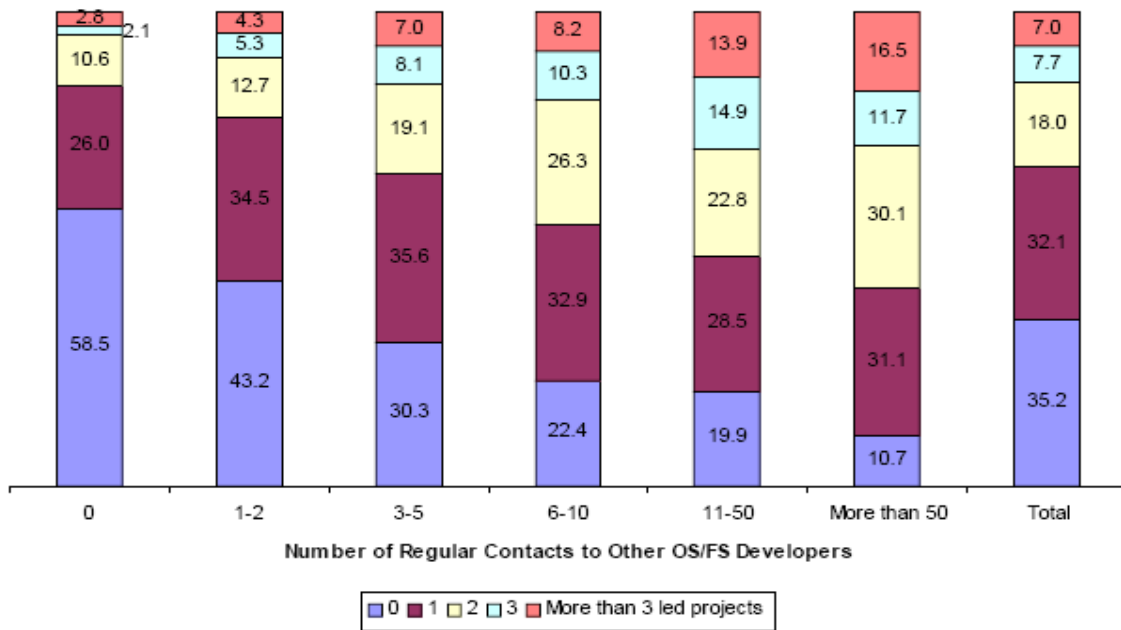
Participating in the FLOSS community is aligned with a high degree of collaboration and communication between numerous people. The FLOSS Developers Survey (Ghosh et al.2002: 40) revealed a bimodal pattern of regular contacts to other community members (Figure 8).



Source: FLOSS Developer Survey 2002

Figure 8: Number of regular contacts of FLOSS community members to other members

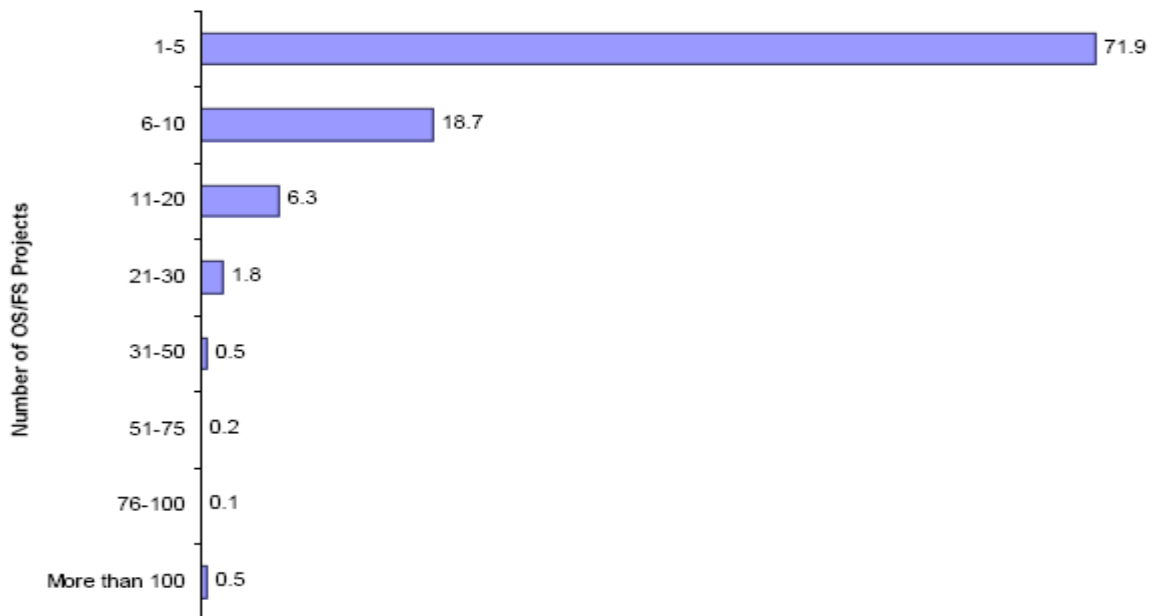
Most community members have regular contacts to 1-5 other community members, 17% have no regular contacts to other members at all, and 15% have regular contacts to 6-10 other community members. Altogether, these shares indicate that the impression of huge volatile networks stretching across the world that people may have when the FLOSS community is imagined is not very realistic. Only a minority of about 5% has regular contacts to more than 50 other community members. Figure 9 shows that this minority obviously consists of community members with a central function (here: leading a project) in large FLOSS development projects or maintainers of forums or mailing lists etc.



Source: FLOSS Developer Survey 2002

Figure 9: Number of regular contacts in the FLOSS community and project leadership

Figure 10 illustrates that a large majority (72%) of FLOSS community members got experience from a rather small number of projects. Roughly one fifth got experience from six to ten projects.



Source: FLOSS Developer Survey 2002

Figure 10: Total number of projects since joining the FLOSS community

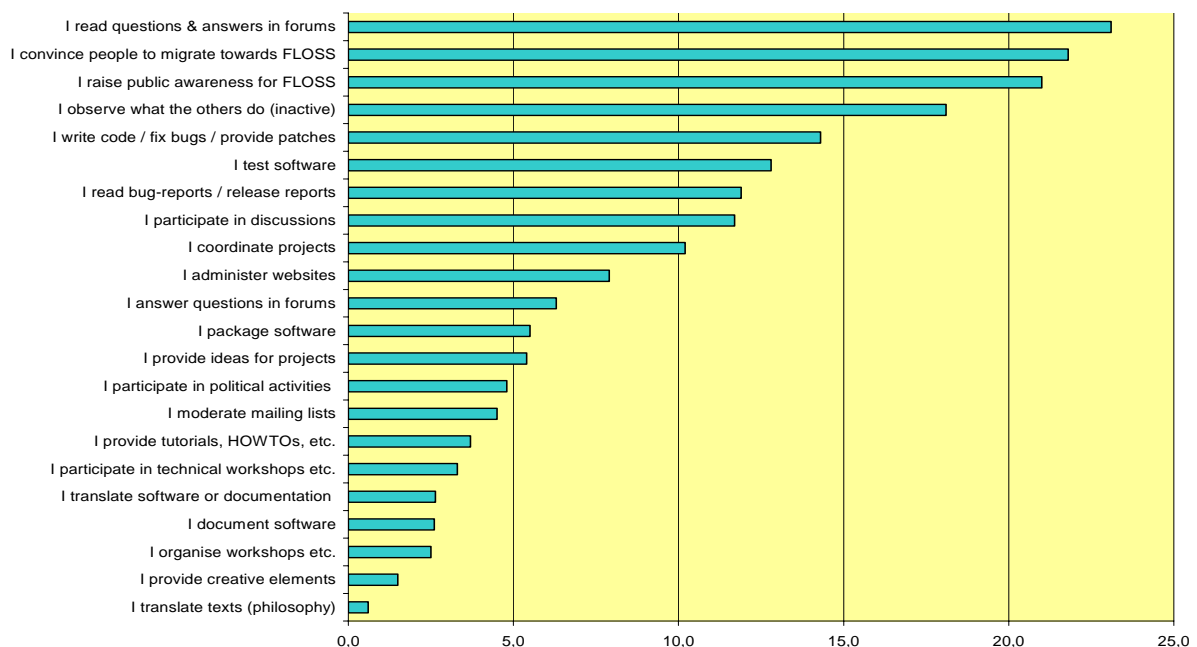
Further insight in the interactions within the community is provided when detailed activities are considered, which has been done in the FLOSSPOLLS Developer Survey (Ghosh & Glott 2005a), which has been answered by 1453 FLOSS community members. The FLOSS community is often described and perceived as a community of software developers. However, it is evident that the activities within the community are not only limited to software development, i.e. to actually coding software. There are people required to translate software and texts, to organise events, and there are a lot of political activities that are clearly not covered by the term software development. By a principal component analysis six distinct fields of activities within the FLOSS community could be distinguished.¹⁵ Each of the activity fields is characterised by a unique set of inter-related activities that distinguishes this activity field from other fields of activity (i.e. an activity that is characteristic for one field of activities does not appear as a typical element of another field of activities):

- Organising /administrating activities
 - I organise workshops etc.
 - I participate in technical workshops etc.
 - I moderate mailing lists
 - I administer websites
- Programming activities
 - I read bug-reports / release reports
 - I test software
 - I write code / fix bugs / provide patches
 - I package software
 - I provide ideas for new features for software projects
- Mobilising (political) activities
 - I raise public awareness for FLOSS
 - I convince people to migrate towards FLOSS
 - I participate in political activities

¹⁵ Together, the six components account for 58 per cent of the variation of answers to the question for the activities of FLOSS developers. In total, respondents could choose from 22 activities.

- Assisting / coordinating activities
 - I provide graphics, sounds, other creative elements
 - I document software
 - I have a coordinating function in projects
- Communicating activities
 - I answer questions in forums
 - I read questions & answers in forums
 - I participate in discussions
- Translating activities
 - I translate software or documentation
 - I translate texts (philosophy)
- Other activities³
 - I provide documents like tutorials, HOWTOs, etc.
 - I observe what the others do

Figure 11 shows the share of community members who perform these 22 items very often.



Source: FLOSSPOLs Developer Survey 2005, n = 1453

Figure 11: Activities performed very often by FLOSS community members

Communicating activities, such as reading Q&A and mobilizing activities provide the main activities within the FLOSS communities, together with the general task to observe what is going on in the community. The second important group of activities is provided by programming activities. The least important activities are to organizing activities and translation, together with some assisting activities.

Based on these fields of activities, a cluster analysis allowed to allocate FLOSS community members to typical activity groups, i.e. groups of community members performing the same distinct set of activities within the community. Overall eight such activity groups can be distinguished:

- Convincers (23% of the community)
- Mobilisers (9%)
- Mobilising communicators (8%)
- High-level activists (5%)
- Assisting mobilisers (11%)
- Indetermined (19%)
- Programmers (13%)
- Programming communicators (13%)

Table 7 shows the activity profile for each of these activity groups across the 22 activities. In order to visualise the distinct profiles of the activity groups only activities that are performed often or very often are considered. The yellow cells indicate shares above average, i.e. activities on which a group is focused. Evidently, the profiles are not clear-cut in the sense that an activity that is typically performed often by one group is not performed often by another group, too. For instance, mobilising activities and programming activities are typical for many activity groups. What constitutes and distinguishes the activity groups is their unique mix of activities across the six activity fields.

		Groups of FLOSS community members by activities								
		Convincers	Mobilisers	Mobilising communicators	High-level activists	Assisting mobilisers	Indetermined	Programmers	Programming communicators	Total
Activity field	Activity	n=255	n=101	n=84	n=58	n=122	n=210	n=149	n=146	n=1225
Organising /administrating	I organise workshops etc.	6,2	38,8	-	50,0	8,3	-	-	7,5	18,9
	I participate in technical workshops etc.	9,1	60,4	7,3	56,1	8,9	9,0	13,5	8,5	19,9
	I moderate mailing lists	22,2	61,3	40,0	67,6	26,5	23,3	22,6	41,1	42,1
	I administer websites	15,7	57,4	12,5	54,2	53,8	13,5	29,6	29,7	36,8
Programming	I read bug-reports / release reports	52,0	42,3	37,8	54,5	36,1	13,9	75,7	51,0	45,6
	I test software	52,2	49,0	45,1	54,5	55,7	8,3	58,5	45,8	45,3
	I write code / fix bugs / provide patches	18,3	47,7	20,6	31,9	49,1	11,8	64,2	51,4	36,4
	I package software	19,5	36,2	22,5	41,5	24,4	-	43,8	28,7	28,9
	I provide ideas for new features for software projects	8,7	31,6	23,1	33,9	32,2	3,9	34,2	30,7	22,9
Mobilising (political)	I raise public awareness for FLOSS	77,3	73,3	68,7	87,9	68,3	22,0	25,2	29,9	54,5
	I convince people to migrate towards FLOSS	79,2	64,0	54,8	82,5	65,6	27,4	20,1	43,1	53,5
	I participate in political activities	21,4	28,4	26,1	58,6	26,6	9,6	10,2	6,2	21,0
Assisting / coordinating	I provide graphics, sounds, other creative elements	3,6	9,3	5,6	12,1	33,6	-	-	2,6	13,7
	I document software	0,8	20,0	15,6	30,8	42,9	3,1	10,4	16,5	16,5
	I have a coordinating function in projects	15,4	67,0	9,7	48,9	62,1	23,4	39,7	43,9	41,5
Communicating	I answer questions in forums	8,6	19,6	37,3	39,3	24,3	4,9	8,3	74,0	23,6
	I read questions & answers in forums	68,9	40,2	81,0	63,8	68,0	53,2	21,9	95,9	61,7
	I participate in discussions	23,8	62,0	48,2	57,9	36,4	12,2	31,8	76,0	38,2
Translating	I translate software or documentation	3,4	3,2	36,1	37,5	5,1	-	6,3	-	14,6
	I translate texts (philosophy)	-	-	8,9	32,1	-	-	-	-	10,6
Other	I provide documents like tutorials, HOWTOs, etc.	7,1	16,1	24,7	43,1	27,7	3,4	9,7	21,1	16,6
	I observe what the others do	73,0	27,6	63,3	46,3	60,7	69,2	43,1	33,9	56,8

Multiple response, percentages do not add up to 100%

Source: FLOSSPOLS Developers Survey

Source: FLOSSPOLS Developer Survey 2005

Table 7: Activity fields and activity groups in the FLOSS community

High-level activists and the two groups of mobilisers (which differ with regard to the role of organizing activities and the residual category of “other activities”) are the most active groups within the FLOSS community, performing the whole or almost the whole spectrum of activities within FLOSS often or very often. The two groups of communicators are also quite active, but they focus on different activity fields and ignore (as compared to other activity groups) organizing and assisting communities. Mobilising communicators are, apart from the undetermined, the only activity group that shows no remarkable activity in the field of programming activities. Programmers have set their focus on programming activities and show no above average share in any activity of other activity fields. Convincers perform some programming activities (reading bug and release reports and testing software) and some mobilizing activities (they like to convince others to migrate to FLOSS or at least try to raise public awareness for FLOSS, but participating in political activities is obviously not on their agenda). Apart from that, it appears very important for this group to keep an overview of what is going on in the FLOSS community in general, as they provide the highest share of those who often or very often observe what the others do. The latter item is the only item where the

undetermined show an above average share.

Interactions between FLOSS community members have been examined by the means of social network analysis.¹⁶ In the management sciences, social network analysis is often referred to as Organizational Network Analysis (ONA), and provides means of making the hidden patterns of information flow of an organization visible. Some benefits of social network analysis studies in FLOSS include the following:

- understanding how FLOSS communities are organized
- interaction between programmers working in a given software project.
- analyzing the structure and modelling the evolution of social interaction in F/OSS communities (Valverde06, Crowston05, Weiss2007).
- studying how software developers and users relate with among themselves, how they are organized, how they take decisions and how they distribute the work load (Barahona2004).
- understanding knowledge collaboration in F/OSS development (Ohira, et al., 2005).
- visualizing how individuals interact in F/OSS projects' mailing lists (Sowe, et al. 2006b), how to find a list of candidates who are expert in a specific topic (Chen, et al., 2006), and seeking out top collaborators and the ways that they collaborate with each other in an organization (Fisher, 2003).

The FLOSS development process produces massive quantities of data which software engineering researchers can use to build and visualize the behaviour and relationships between individual nodes. Such large source of data gives researchers an unprecedented level of detail in social network analysis, along with the potential for new understanding, and useful predictions. In recent times, social network analysis studies in FLOSS have intensified. In Xu, et al., 2005 and Madey, et al., 2002, software developers form nodes on a network (graph) and links or edges exist between the nodes if developers participate on the

¹⁶ The field of social networks makes the observations and measurements of these relationships possible (Sowe, 2007d; Denning, 2004). A social network is a social structure made of nodes (which are generally individuals or organizations) that are connected or linked by one or more specific types of relations or ties, such as belonging to the same project or community. A link can either be directed or undirected. A direct link may exist between two nodes A and B if there is one-way relationship. That is A talks to B. An undirected link, on the other hand, shows a reciprocal relationship of the form A talks to B and B talks to A. Social networks are usually represented as graphs, depicting relationships or connections that exist between the entities being studied. For instance, Figure 12 is a social network showing the relationship between individuals and three. In the figure, mailing lists form nodes (represented by squares) or points on a bipartite graph and two or more nodes are linked by a line if a poster (represented by spheres) posts email messages to the lists. The abundance of data resulting from the FLOSS development process and individuals' activities in the Internet has made it possible for researchers to observe and measure millions of linkages or relationships that may exist between individuals. The internet enables virtual interaction that overcome the barriers associated with face-to-face communication between individuals, some researchers (Steven and Faraj, 2005) have coined the term "Electronic knowledge networks" or (EKNs) to refer to network of individuals who rely on electronic communication to exchange knowledge.

same project. In these studies, identification of clusters exposed connected groups of software developers across FLOSS projects hosted at the sourceforge.net. Such networks revealed a small group of highly prolific software developers or "linchpins". In a study of mailing list network, Sowe, et al. 2006 found out that participants in one list are connected to participants in another list in the Debian project (Figure 12). The presence of knowledge brokers in the mailing lists network in is akin to the discovery of linchpin developers in Xu, et al., 2005 .

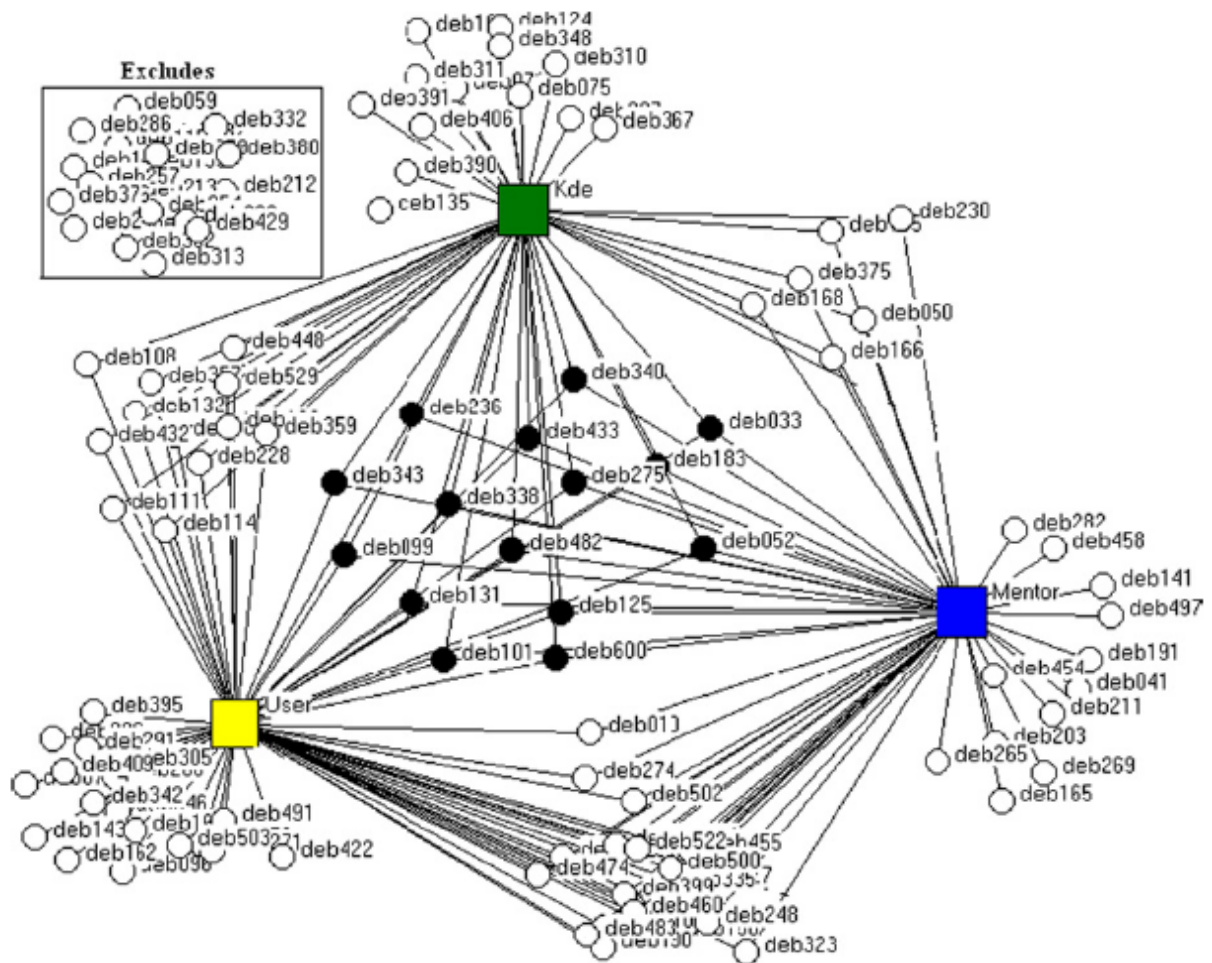


Figure 12: Knowledge brokers in mailing lists network

The visualization (Figure 12) shows that posters (community members who post in a forum) could be located at more than one node in the network and are capable of sharing their knowledge with other participants in other nodes. The exchange of email messages was used to establish ties between the nodes (Ghosh, 2004; Hanneman and Riddle, 2005). Each tie was weighted as the number of email messages contributed by each individual. The mailing lists network enabled the identification of three groups or structures within the lists.

1. **Group one.** Consider, for example, posters *deb367* in the KDE¹⁷ list, *deb141* in the Mentors list and *deb422* in the User list. These posters posted more information and/or provided more assistance within the lists they participated. Both knowledge seeking and knowledge providing are localized in these groups.
2. **Group two.** Posters who participated across lists, externalizing their knowledge and expertise and helping answer questions knowledge seekers posted. For example;

User U KDE = {deb111, deb114,...}

KDE U Mentors = {deb230, deb050,...}

MentorsU User = {deb010, deb274,...}

3. **Group three.** At the center of the visualization is a group of 15 posters (e.g. *deb600*) called *Knowledge brokers* (Sowe, et al. 2006b). They link all the three nodes in the network. By linking and collaborating with list participants, they serve an important function as community facilitators or hubs (Barbasi and Bonabeau, 2003) on the network. Barbasi, 2002 (pp. 64) argued that hubs are special and paying attention to them is well deserved. They are active members and dominate the structure of all networks in which they are present, creating trends and fashions, making important deals, etc.

In a similar study using mailing lists data to study the social network structure behind the World Wide Web Consortium (Chen, et al, 2006) also found few experts located in the center of the network of the communities in the W3C mailing lists. Thus, few conclusions can be derived on the roles of experts or knowledge brokers in F/OSS projects mailing lists:

- These experts or knowledge brokers are surrounded by a lot of other users- "peripheral users"
- Knowledge brokers replies more often to questions posted to mailing lists than peripheral users. As with the case of the W3C mailing lists, on average, candidate experts reply more often than non-candidate users.
- Knowledge brokers do not cluster into some densely linked groups (or cliques) in mailing lists networks. Probabilistically, they are usually separated into some communities with other mailing lists participants.

¹⁷ KDE is a desktop environment for UNIX workstations, similar to those found under the MacOS or Microsoft Windows.

A number of other studies have also employed social network visualizations to visualize communication flow among individuals by email logs (Gloor, et al., 2003}, to study interaction among IRC channels users (Mutton, 2004), and to visualize the relationships among modules in the Apache project (Barahona, et. al., 2004). Thus social network analysis offers an important mechanism for the discovery of key individuals in FLOSS communities, and help us determine *who is who*, who is interacting with who, how much individuals are contributing to community discourse, etc.

4.2. Learning Resources (non-technical)

FLOSS communities provide users with various types of learning resources, the “common” ones like manuals, tutorials, or wikis, but also resources that might not be considered as learning resources at a first point like mailing lists and forums. One unique aspect of all of them is that they are jointly generated by user and developer and after generation continuously updated and improved.

Scacchi (2002) identified 8 types of what he defined as “software informalisms” with each having sub-types. With his 8 type classification Scacchi provides an overview of the information systems within the FLOSS communities and their purposes. According to Scacchi these software informalisms are used for (software) product requirement definition, sense making, continuous discourse, and accountability. As noted by Scacchi the requirements for a FLOSS product are, unlike for traditional software products, not pre-defined, but specified through developer and user discourse that reference or link:

- “email or bboard (forum) discussion threads,
- system vision statements,
- ideas about system functionality and the non-functional need for volunteer developers to implement the functionality,
- promotional encouragement to specify and develop whatever functionality you need, which might also help you get a new job, and
- scholarly scientific research publications that underscore how the requirements of domain-specific software (e.g., for astronomical imaging), though complex, are understood without elaboration, since they rely on prior scientific/domain knowledge and tradition of open scientific research.”(Scacchi 2002)

From the learning point of view the 8 informalisms might help in understanding the type of “Learning Resources” that users in FLOSS in general dispose about. Following it will be briefly referred to the each informalisms, to subsequently have a look at them from a “Learning Resource” point of view.

The software programme itself might be seen as an analogue to the content of a course in formal education. But unlike in education, or even in formal software development, there is no such think as a “Requirement Specification” document for FLOSS products. Instead users and developer are in a constant re-negotiation of the software’s features, functions or design (Scacchi 2002).

“...it appears that the requirements for open software are co-mingled with Design, implementation, and testing descriptions and software artifacts, as well as with user manuals and usage artifacts (e.g., input data, program invocation scripts). Similarly, the requirements are spread across different kinds of electronic documents including Web pages, sites, hypertext links, source code directories, threaded email transcripts, and more. In each community, requirements are described, asserted, or implied informally. Yet it is possible to observe in threaded email/bboard discussions that community participants are able to comprehend and condense wide-ranging software requirements into succinct descriptions using lean media [39] that pushes the context for their creation into the background. (Scacchi 2002)

Community communications

Mailing lists and forums are the common place for community communications to discuss about the requirement of the software or known bugs, but also other organizational aspect and they are also the main place to provide support to users. Chats, instant messagings or voip are also used but for more ad-hoc discussions. The advantage of communications in mailing lists and forums is that other user can later on read through these.

In FLOSS messages often do not consist of questions and answers only, but also include the “path” leading to the answers. This might include parts of the code discussed together with references or links to other messages or software websites. Following Scacchi this provides “some sense of context for how to understand messages, or where and how to act on them.”

Hemetsberger & Reinhardt (Hemetsberger & Reinhardt 2006a) suggests that members of FLOSS communities learn and build collective knowledge through the use of ‘technologies’ and the establishment of discursive practices that enable virtual re-experience. Following the

problem solving processes, or other type of argumentation lines, are important learning resources of FLOSS communities that enable other users' re-experience. With this users get access to "knowledge that is often tacit in nature but visible and observable in the common practice of and interactions among competent practitioners", which is "also highly contextual and, therefore, cannot be externalized and taught independently from its context." (Brown & Duguid 1991)

Community members are well aware of the role of mailing lists and forums and consequently expect that these resources are used first, before individual support might be provided .

Scenarios of usage as linked Web pages

To explain the functioning of the software "community participants create artifacts like screenshots, guided tours, or navigational click-through sequences (e.g., "back", "next" Web page links) with supplementary narrative descriptions in attempting to convey their intent or understanding of how the system operates, or how it appears to a user when used...participants may publish operational program execution scripts or recipes for how to develop or extend designated types of open software artifacts." (Scacchi 2002). As a use case live demo versions are also commonly available where users can log in at the front and back-ends to experience the software in practice.

HowTo Guides

How to guides are also provided that explain how the software functions. Additionally communities might dispose about FAQs, knowledge bases or wikis. (Meiszner 2007). Further valuable "How To Guides" are also the community forums.

External Publications

All of the 4 cases that were examined by Scacchi, and 50% out of the 80 communities that were reviewed by Meiszner (2007) provided also external publications. These publications might consist of technical articles, books, news feeds, blog postings, or professional and academic articles.

"Academic articles that are refereed and appear in conference proceedings or scholarly journals...serve a similar purpose as professional articles, though usually with more technical depth, theoretical recapitulation, analytical detail, and extensive bibliography of related efforts. However, it may be the case that readers of academic research papers bring to their reading a substantial amount of prior domain knowledge. This expertise may enable them to determine what open software requirements being referenced may be

obvious from received wisdom, versus those requirements that are new, innovative, or otherwise noteworthy.”(Scacchi 2002)

Open Software Web Sites and Source Webs

Community websites have the advantage to provide the community with an information infrastructure for “publishing and sharing open descriptions of software in the form of Web pages, Web links, and software artefact content indexes or directories. These pages, hypertext links, and directories are community information structures that serve as a kind of organizational memory and community information system. Such a memory and information system records, stores, and retrieves how open software systems and artefacts are being articulated, negotiated, employed, refined, and coordinated within a community of collaborating developer-users” and it might “include *content* that incorporates text, tables or presentation frames, diagrams, or navigational images (image maps) to describe their associated open software systems. This content may describe vision statements, assert system features, or otherwise characterize through a narrative, the functional and non-functional capabilities of an open software system...Web content that describes an open software system often comes with many embedded Web *links*. These links associate content across Web pages, sites, or applications.”(Scacchi 2002)

A characteristic of FLOSS is also the access to its source code. The source code can be usually accessed at the project’s website or at repositories like e.g. sourceforge.net.

Software bug reports and issue tracking

“One of the most obvious and frequent types of discourse that appears with open software systems is discussion about operational problems with the current version of the system implementation. Bugs and other issues (missing functionality, incorrect calculation, incorrect rendering of application domain constructs, etc.) are common to open software, much like they are with all other software. However, in an open software development situation, community participants rely on lean communication media like email, bug report bboards, and related issue tracking mechanisms to capture, rearticulate, and refine implicit, mis-stated, or unstated system requirements”(Scacchi 2002).

Traditional software system documentation

Open software systems are not without online system documentation or documentation intended to be printed in support of end-users or developers. Scacchi noted that a general problem of this type of documentation is that it is usually outdated, but that FLOSS software

documentation, unlike the documentation of traditional software, benefits from the fact that it can be updated by the community (user/learner generated content).

This could explain why many FLOSS communities introduced wikis (Meiszner 2007) as this might be more convenient for constant changes and updates.

Software extension mechanisms and architectures

“The developers of software systems in each of the four communities seek to keep their systems open through provision of a variety of extension mechanisms and architectures. These are more than just open application program interfaces (APIs); generally they represent operational mechanisms or capabilities.”(Scacchi 2002)

4.3. Knowledge creation in FLOSS communities

Hemetsberger and Reinhardt (Hemetsberger & Reinhardt 2006a; Hemetsberger 2006b) examined the learning and knowledge creation processes within FLOSS communities at the individual and group level. The objective of their work was to understand “how knowledge sharing and creation processes develop at the interface of technology and communal structures that effectively exploit the advantages of Internet technology and at the same time are able to overcome the problem of tacit knowledge transformation.”

For their study Hemetsberger and Reinhardt adopted a social view of learning and knowledge creation that promotes the idea that knowledge is deeply embedded in the technological and social context of a community that creates and reproduces knowledge.

They view learning in FLOSS communities as experiential learning whereas learning is a process whereby learning is created through the transformation of experiences as developed by Kolb (1984).

In conceptualizing ways how to enable sharing and creating knowledge online Hemetsberger and Reinhardt, like Bacon and Dillon, drew “on the communities of practice literature (Lave and Wenger 1990; Brown and Duguid 1991; Wenger 1998; Wenger 2000) and on Donald Schön’s notion of ‘the reflective practitioner’ (Schön 1999).” Hemetsberger and Reinhardt carried their work out alongside three questions:

1. How do community members organize content with regard to their daily routines that potentially transforms into knowledge for other members?
2. As open-source communities depend on attracting and socializing new members: how

are new members enabled to accumulate the knowledge necessary for becoming a valued member?

3. How do members co-create and conceptualize new ideas – create new knowledge – in absence of physical proximity?

One of their key findings and a valid answer to all of these three questions, as highlighted at the following paragraph, is: “enabling re-experience”: Enabling re-experience is a crucial requirement for online learning and knowledge-building and a principle that FLOSS communities are following.

“Our findings make one key assertion. They reveal that *enabling re-experience* constitutes the fundamental mechanism for learning and knowledge-building to occur online. From an individual perspective, learning is initiated by displaying information-rich content, both in a structural and in a sequential order, as well as by instructive content and discourse. At the collective level, we found participative practice, collective reflection and virtual experimentation processes to be explanatory with respect to knowledge-building. Knowledge manifests itself online through a variety of contents displayed, as well as through online discourse. Our analysis further documents how those manifestations of knowledge initiate individual and collective processes of learning and knowledge-building. In the following those processes are described in detail.” (Hemetsberger and Reinhardt, 2006a)

Re-experience is assured at various levels and by the use of different tools as following briefly explained. Hemetsberger and Reinhard reviewed FLOSS communities from two perspectives: the Learning Community and the Knowledge building community, describing for each perspective where and how re-experience is assured.

The Learning Community

Past experiences and the problem solving processes are preserved and made available for others at:

- the source code; in the format of comments
- the CVS repository and change log
- the archives at the mailing list and forums
- the documentary and / or wiki

As an illustration Hemetsberger and Reinhardt refer to the KDE projects website where that is clear and transparent structured providing “information-rich content, in combination with hypertext technology and the implementation of search functions”, which is seen as being “of key importance for initiating productive *inquiry* and reflective observation”. Another

important pre-requisite is open access to content and communication channels in order to initiate search, individual reflection processes, and active experimentation.

Learners are provided with training facilities. To their surprise Hemetsberger and Reinhardt found that the used technological tools are relatively simple, but still allow learning to take place without person-to-person interaction, though in the case required help is available through tools like e.g. chats or forums.

The existence of training facilities is, according to Hemetsberger and Reinhardt, not a replacement to mentorship and / or face-to-face contacts for learners at the very beginning stage. At this stage mentorship and/or face-to-face contact still seem to outperform Internet technology and computer-mediated communication.

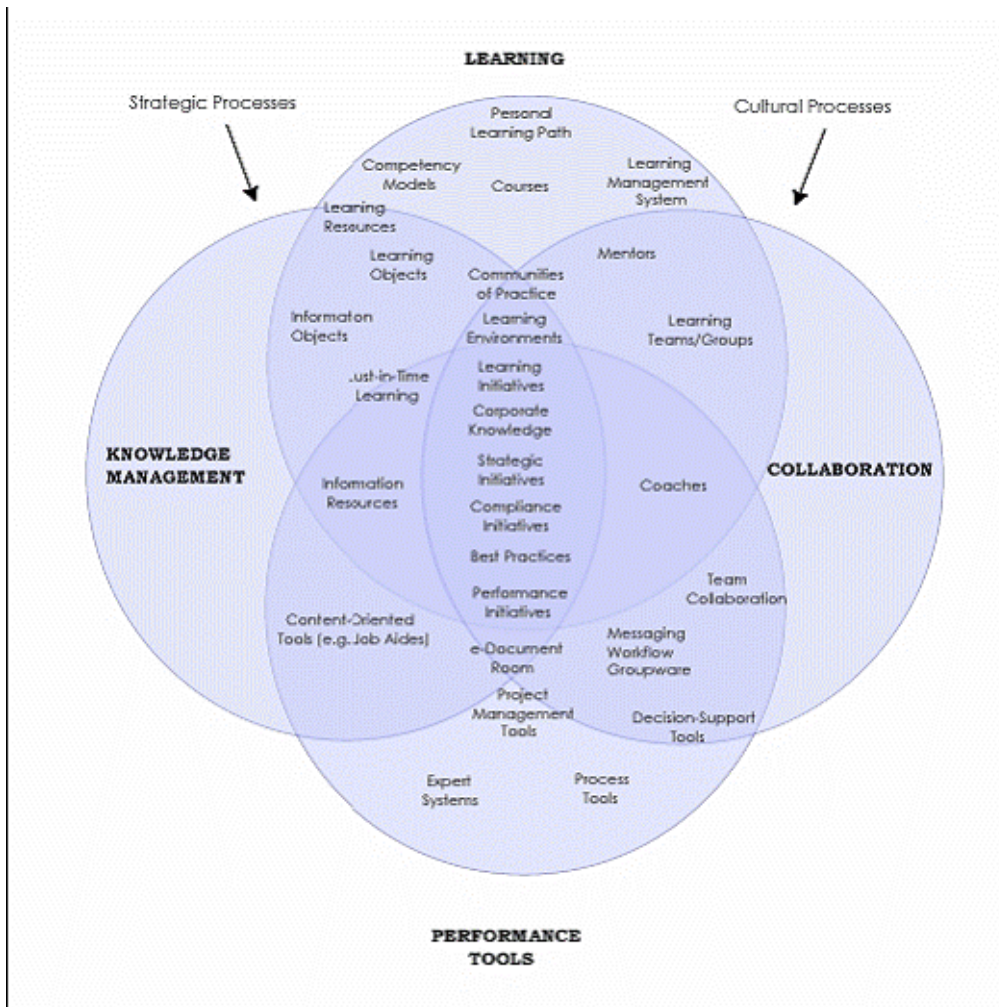
The Knowledge-building Community

Participative practice is assured through the CVS system and log file that keeps members automatically updated on changes and e.g. allows for quick scans through changes. This also includes information on the type of changes and the reasons behind.

This is assured through mailing lists or forums as platforms for members to engage in communication and reflective discourse. The virtual environment, the different (technological) components, the design of processes and structures play a crucial role for the coordination of activities and to create and maintain a shared framework.

The findings of Hemetsberger and Reinhardt are well in line with the ones of Lin (Lin 2005a; Lin 2005b). Lin approaches FLOSS from a social world's theory viewpoint and justifies this with the diversity of knowledge and layers of social organisations, dynamics and negotiation orders within the FLOSS communities, which is higher than at other communities with more firm boundaries.

Her view is "that FLOSS knowledge creation not only exists in code writing (i.e. converting tacit knowledge into explicit knowledge as previous studies usually presuppose), but also in everyday problem solving activities (tacit knowing is embodied and embedded in solving problems)." She also highlights the four main factors relevant for learning in FLOSS (Figure 13): the applied performance tools as the basic requirement that enable knowledge management and collaboration, which by the end supports the learner in his learning process.



Source: Lin 2005

Figure 13: Beyond the CoP (Community of Practice) Strategy

The focus of Lin’s work seems to be on two points: problem solving and the importance of tacit knowledge. As Lin points out, problem solving is “considered as the most essential practice that evokes interactions. Knowledge creation in FLOSS communities is embodied and embedded in solving a problem.” According to Lin “this problem-solving process is not a straightforward ask-and-answer one-way activity. It involves various heterogeneous and contingent factors concerning cognition, social skills and technical expertise.”

Like Hemetsberger and Reinhardt, Lin also stresses that the problem solving process, that involves much of tacit knowledge, becomes explicit through engagement in online discussions and the creation of new knowledge. Due to the application of appropriate technologies (CVS, mailing lists, forums, wikis, etc.) the social interactions become and remain visible and therefore can support others within their learning activities. Thus the problem solving process of one or more individuals is a learning activity for the individual

that leads to the creation of new knowledge.

However, for knowledge creation purposes, asynchronous communication technologies, such as discussion forums and mailing lists are used in order to make community members think before they act and respond. In order to be successful in that respect, a strong culture of freedom, openness, and helpfulness has been promoted throughout the history of the community. (Hemetsberger, 2006b)

Within her work Lin shows by the mean of cases “how conversations can be enacted via engaging diverse actors in the problem solving processes, and how (tacit and local) knowledge can be transmitted when problem based locality is shared.”

4.4. Knowledge exchange in FLOSS communities

“Software development is a knowledge-intensive activity that often requires very high levels of domain knowledge, experience, and intensive learning by those contributing to it” (von Krogh *et al*, 2003). As stated in chapter 1, FLOSS communities are an exemplar of successful software development. Moody (2001), Raymond (1999), Sawhney and Prandelli (2000) and Wayner (2000) suggest that this success is related to the growth in the size of the developer community. Hence, the joining of new members (newcomers) is extremely important in order to maintain the sustainability of FLOSS projects. This, in turn, increases complexity, since these newcomers have to be culturally integrated and taught in order to help them to become competent members. At the same time, growing size of the community also aggravates spreading the overall knowledge of the community to the individual community members. Therefore, to offer opportunities for learning and self-development for newcomers as well on the horizontal level between the experienced (but ever-learning) community members, FLOSS communities have to define rules and norms, roles and facilities, i.e., they have to provide channels in which knowledge is transferred from one community member to another and from the experienced to the newcomers who are interested in contributing.

Regarding the learning process of newcomers, one could think of mentorship as a means for knowledge flows from the experienced to the newcomers. Mentorship is an informal relationship between an expert (mentor) and a newcomer (mentee or novice) for the purpose of learning and self-development. However, mentorship in FLOSS communities appears to be impossible because of the community dynamics, especially its fast growth in recent years. If learning for newcomers would solely rest on the hands of experts the community would probably not have been able to cope with the massive inflow of newcomers since 1990. One should recall in this context that, according to the FLOSS Developer Survey (Ghosh *et al*.

2002), in 1990 the size of the community was only 8% of its size in 2002 (see Figure 1). Furthermore, it should be considered that young novices obviously need about 5 years to become experts (see table 5). Together, these two observations imply that even under the improbable assumption that each member of the community in 1990 was an expert each of these experts would have been obliged to be a mentor for 5-6 or even more newcomers until 2002. Developing software and contributing to the FLOSS community as a social movement would probably not have been possible to the observed degree if the learning process in the community would rely on mentorship. Another reason that militates against mentorship is that “experts may lack patience to guide a novice, and, from the novice’s viewpoint, someone more proximate in experience may be a better teacher than the expert because the knowledge gap is not as great” (Swap *et al*, 2001).

Therefore, other community members who are just advanced learners also support the knowledge transfer process by transferring their knowledge to community members that have *marginally* less knowledge. *Conclusively, what characterizes knowledge transfer within FLOSS communities is that this transfer can take place between community members that have only marginal differences with regard to their expertise.* As will be shown in chapter 5, the technology that is used within the FLOSS community in order to communicate and facilitate knowledge exchange enables knowledge transfer *between community members that do not know each other (personally) and / or that live in completely different places of the world.* These *marginal knowledge differentials might be punctiform*, i.e. someone who belongs to the community for many years and with a lot of expertise in many fields can learn from a newcomer if the newcomer disposes of a specific knowledge in a specific field that is unknown to the expert.

Practically, the first learning steps are the most difficult because newcomers usually do not know what questions to ask. So, many experts strongly recommend first reading other’s code, the software manual (RTFM – “Read The Friendly Manual”), the list of frequently asked questions (FAQ) or searching in the web (STFW - "Search The Friendly Web"). Moreover, usually the community doesn’t give support on things that could be easily solved once performing one of these activities. The following excerpt, taken from the KDE-devel-mailing list (Reinhardt and Hemetsberger), provides an example of RTFM:

Newcomer:

Hello Everyone,

I am totally new to KDevelop, please let me know what it is, when I saw it for the first time I found it as if it is for developing new applications for KDE ...

Please tell me how to start with KDevelop ... if I want to develop some Applications like what we do with visual basic on Windows platform then what is the best (Let me know whether I can do something with QT Designer for Developing Applications to run on Linux ...)

Anyone please help me in this regard ... I am very much interested to develop GUI applications for Linux ...

Thanks & Regards

Expert:

Maybe you want to wait for Visual Basic for Linux? Perhaps it is available in about 50 years. Since you are new to Linux I can give you the astonishing advice to read the documents which come with your system or are available at <http://www.kde.org/>. I do not believe that someone here has the time to write the documentation especially for you.

I do not know where you come from. Perhaps you are used to Win systems. Obviously new users there get a short introduction to system and all software packages by Bill himself.

You can believe me that I do not expect this mail to help you, but I could not resist.

Sorry!

However, when the newcomer shows that he has some degree of knowledge on the subject and that he has tried to fix the problem, the community provides gratis support. The following excerpt, taken from the Apache Usenet (Lakhani and von Hippel, 2003), presents an example of this service:

Newcomer:

Subject: Apache 1.3.1 and FrontPage 98 extensions.

A small problem . . .

Hi,

I've compiled and installed Apache 1.3.1 with mod frontpage.c. That section seems to be working. I have installed the FrontPage 98 extensions, and that seems to almost be working, but I can't find any relevant information anywhere about how to solve this problem. I can look at a home page for a user, but I can't publish to it. Whenever Front-

Page tries to connect to the server, this message appears in the error logs:

Thu Oct 8 10: 13:31 1998 [error] (104) Connection reset by peer: Incorrect permissions on webroot "/usr/local/httpd/htdocs/_vti_pvt" and webroot's _vti_pvt directory in FrontPageAlias().

Thu Oct 8 10: 13:31 1998 [error] File does not exist:

/usr/local/httpd/htdocs/vti_bin/shtml.exe/_vti_rpc

I haven't a clue how to fix it. Any help will be very appreciated, and a reply by e-mail will be noticed more quickly (I'm terrible at remembering to check the newsgroups)

Thanks!

Expert:

Hi there,

There are two possible causes of your problem:

1. Make sure owner and group are the same and that the directories have the same set of permissions. /home/user/public_html user group/home/user/public_html/_vti_bin www group1 should be: /home/user/public_html user group/public_html/_vti_bin user group

2. Apache-fp utilizes fpexe and mod frontpage to provide a higher level of security. Part of the mod frontpage code sets LOWEST_VALID_UID and LOWEST_VALID_GID. Users with UIDs and GIDs less than these values will not be able to run the server extensions. These values are configurable. For more information please check the SERK documentation and the Apache-fp page.

Greetings.

These examples also illustrate a procedure which is commonly referred in the literature as “pulling” (Berends, 2002), i.e., a learning process where it is the newcomer who takes the initiative, for example, by asking a question to a more advanced learner. Otherwise, if it's the advanced learner who takes a proactive role and preselects the relevant and necessary information for the newcomer, this process is called as “pushing”. Billet (1994) suggests four different guided-learning strategies by which a mentor can support a newcomer:

1. **Modeling:** involves the mentor performing a task, which the newcomer observes. An example of “modeling” may be the code tracking in the KDE subversion (SVN) repository, using colors, tables and comments that describe the changes made;
2. **Coaching:** consists of the mentor observing and monitoring the newcomer's

performance, providing guidance, feedback, hints and tips;

3. Scaffolding: involves the mentor providing the newcomer with opportunities to acquire knowledge that is within newcomer's capacity. An example of "scaffolding" may be the provision of tutorials, with exercises on coding and a step-by-step guidance of how to understand these exercises. Thus, tutorials are also a very important tool to support the newcomers first steps;
4. Fading: consists of "gradual removal of support until learners are able to conduct the task autonomously" (Billet, 1994, p.12).

As stated in Lakhani and von Hippel (2003), the mentorship is not assigned, but is based on volunteering. These authors concluded this by asking to a sample of 21 mentors (information providers) to express their agreement or disagreement regarding the motivations for providing answers to help the newcomers (information seekers) on Apache Usenet (Table 8). The results of the statement (9) "because it is part of my job" make it clear, with 63% expressing disagreement, 27% expressing neutrality and only 10% expressing agreement. Other statements that are consistent with this conclusion are the three affirmations (1)-(3) in Table 1 having to do with reciprocity: "I help because I have been helped and/or expect to be helped in the future". Moreover, the majority of the information providers reported that they did not know the information seeker they were helping.

I was motivated to answer because	Mean (standard deviation)
(1) I help now so I will be helped in the future	4.52 (1.25)
(2) I have been helped before in CIWS-U ¹⁸ — so I reciprocate	4.85 (2.08)
(3) I have been helped on Usenet before — so I reciprocate	4.61 (1.96)
(4) I answer to enhance my career prospects	3.76 (1.55)
(5) I want to enhance my reputation in OSS/Apache community	4.71 (1.35)
(6) I answer because its fun	4.81 (1.44)
(7) I answer to promote OSS	5.14 (1.35)
(8) I answer to take a break	4.65 (1.65)
(9) I answer because it is part of my job	2.23 (1.76)
(10) I have expertise in this area	4.47 (1.32)
(11) I am the authority in this area	2.47 (2.14)
(12) I answered because I thought the poster would not get a good answer if I did not	4.52 (1.57)

seven-point scale: 1 - strongly disagree, 7 - strongly agree

Table 8: Providers’ views regarding their motives for providing answers to help information seekers on Apache Usenet

In addition, when the same authors asked to the information providers the main motive for reading or scanning questions on Usenet, the responses indicated to learn about problems that other Apache users are experiencing (Table 9). In other words, this learning helps them to control and improve their own Apache websites and software code.

¹⁸ CIWS-U is one of two Usenet newsgroups that address questions related to Apache web server software.

Reasons for reading	Mean (standard deviation)
To learn	5.90 (1.58)
To answer	4.95 (1.02)
For fun	4.29 (1.19)
For break	4.81 (1.33)

Table 9: Providers' reasons for reading or scanning questions on Usenet.

Hence, as stated also in Schon (1987), the learning process is not only important for the newcomers, but also for the mentors, because it involves some form of reflection-in-action. More precisely, as the newcomers may have different perspectives and may come up with pertinent questions, the mentor is forced to reflect, possibly causing a rethinking of taken-for-granted assumptions and consequently an improvement in the learning process.

5. TECHNOLOGICAL LEARNING RESOURCES

FLOSS projects are almost exclusively administered online. One of the most important prerequisites for coordination and cooperation on the Internet is provided by the functionality of various communication and groupware tools. They provide a meeting place for online interaction without regard to time or physical location (Hemetsberger 2006). Many of these tools were built by Free/Libre Open Source Software (FLOSS) communities; creating products like wikis (Wikimedia), blogs (Wordpress), or social networking solutions (Elgg). This section is aimed at providing some information on the type of tools that are available and used within the FLOSS communities.

FLOSS projects are either hosted at their own platforms and systems, or they host their project at a repository like sourceforge.net, or they choose a mixture of both. A mixture of both has the benefits of creating a tailored community environment with own identity at the own platforms/systems and to still take the advantage of the web based collaborative revision control and software development management systems that sourceforge.net and others offer. It might be required to shortly address the role of Sourceforge.net as being one of the (if not *the*) largest providers with currently over 140.000 registered projects and more than 1.5 million registered users. Thus the benefit of placing a project partly or total at a side like sourceforge.net lays not only within the provided services, but also within the visibility a project gains. Systems like sourceforge.net provide FLOSS projects with a number of tools from the day of start – even without having a own platform or system. Sourceforge e.g. provides projects with a project website, a subversion system (SVN which can be compared with the concurrent versioning systems / CVS), mailing lists, forums, repositories, or a documentation space.

In a review of FLOSS projects that are listed at the www.opensourcecms.com[2] website Meiszner (2007) found out that from 113 reviewed FLOSS projects 80 disposed about a own platform or system (71%), meanwhile the remaining 33 (29%) were hosted at a service like sourceforge.net. Though all of the reviewed communities develop software that provides a broad range of communication and information tools, the tools that were actually used within these communities appeared to be rather narrow as following shown. Almost 94% of all communities had a forum and Documentation/Knowledgebase available and in 51% of the cases a wiki was used for documentation / knowledgebase purposes (Table 10). Further on more than half of the projects provided some type of news services with “85% of the communities offered a project related news section, with another 50% offering additional

news such as on the latest forum posts, latest blog entries, or RSS news feeds. Tags were used in only 5 (6%) and video / podcasts in only 4 (5%) communities” (Meiszner 2007) .

Tools for knowledge development and sharing		
	n	%
Forum	75	93,8%
Dokumentation / Knowledgebase	75	93,8%
News (Flash)	68	85,0%
Wiki	41	51,3%
Latest News Various	40	50,0%
Blog	27	33,8%
Tags	5	6,3%
Discussion page	4	5,0%
Video / Podcasts	4	5,0%
n = 80		

Source: Meiszner 2007

Table 10: Tools for knowledge development and sharing

Relationship and trust building tools as described by Pór (Pór, 2004) are in most cases integrated into the members profile within the forum software (and only occasionally outside of the forum).

The high number of communities with member profile features, as detailed at table 11, might be explained due to the fact that most of the used forum solutions are existing software solutions that provide these features already “on-board” (Meiszner 2007) .

Relationship and trustbuilding tools		
	n	%
Profiles	76	95,0%
Within the profiles option:		
Members Roles / functions / Groups	66	82,5%
Members publications, posts, etc	63	78,8%
Members MSN, skype, chat, etc. information	60	75,0%
Members preferences & interests	37	46,3%
Geo Map	6	7,5%
Buddy list	13	16,3%
FOAF	2	2,5%
My Tasks	1	1,3%
Own(wiki)page	8	10,0%
Outside of the profiles option:		
Calendar / Events	20	25,0%
Chat	19	23,8%
Polls	18	22,5%
"who is online"	46	57,5%
"who sees what"	3	3,8%
Shoutbox	8	10,0%
n = 80		

Source: Meiszner 2007

Table 11: Tools for relationship and trust building

In 82% of the cases the profile provided information on the members' role, its hierarchy and functions within the community, being calculated e.g. upon the number of postings, responses, ratings or solved problems. Another 79% of the cases also provided information on the members' latest posts, publications, or bloggings.

In 75% of the cases, members also had the option to provide information on their VOIP and messenger accounts and at 46% of the communities they could provide information on their interests or preferences. Though being in general available Meiszner also noted that a brief scan through each community showed that many members did not seem to use this option of providing these type of information.

As detailed by Giuri (Giuri 2004) FLOSS communities seem to rely on two important infrastructures: modular design and the use of Internet. Internet (email, newsgroups, forums etc.) reduces transaction and communication costs among developers and therefore provides a

fundamental infrastructure for distributed development across space and over time.

The underlying technological infrastructure that can be found in FLOSS communities is both simple and mature. Code is usually stored in CVS, documents and manuals in knowledge bases or wikis, and additional information are published through the project's website, newsletters, or blogs (Hemetsberger 2004). Communications occurs mainly asynchronous through mailing lists or forums and are therefore preserved and available for the entire community. The availability and integral application of this diverse range of tools is one of the most important preconditions for collaboration and enable mass participation in collective activities as they can be found at FLOSS communities (Hemetsberger 2006b)

The KDE community, which was investigated by Hemetsberger (Hemetsberger 2004), provides an overview of how a technological community infrastructure might be. In order to be able to digest the huge amount of knowledge and information and to build up a group memory, knowledge technologies and task-related features are implemented that decrease complexity. "This is, for instance, the modular structure of tasks, keeping track of code in a CVS repository, and shifting the locus of knowledge from individuals to a transactive group memory where members know *where* to find information.

For example: emphasizes homogeneity. Both concepts reflect some (though, not all) facets of team knowledge, and both are said to have positive impact on team's performance.

To foster comprehension developers also add comments to their source code (reflection-on-action) which enables a re-thinking and re-experiencing process among the other community members." Following Hemetsbergers the "most important building block of the KDE community's knowledge system consists of 81 mailing lists" for discourse and open reflection and as an archive for transactive memory of the learning community."

Analogue to the mailing lists the forums are equally important. Meanwhile mailing lists are targeting the developer community; forums target the community at large (e.g. the user). These asynchronous communication technologies are not only valuable for knowledge creation purposes, but also in order to make community members think before they act and respond (Hemetsberger 2006b).

Though still not answered in detail there are indications for different preferences of the different community groups. As an example it is referred to two case studies in this domain one on the Freenet community (Krogh, 2003) and another one on the Apache community (Hippel, 2002). The first study focuses on the core team and suggests that mailing lists are the

main way of communication within the community, meanwhile the later one examines the support environment and providing the impression that the main way of communication is based on forum posts.

Table 12 provides a general overview of the different types of tools that are used within FLOSS communities and the way they impact learning (Hemetsberger 2006a). Wikis, documentation and knowledge management systems are also important for co-authoring and collaborative content management (Pór 2004) and would (depending on their usage) belong to the transactive group memory or instructive content.

Displays and manifestations of knowledge	Technological tools	Initiated processes of learning and knowledge building
<i>Code</i>	CVS repository	Full cycle of re-experiencing: Concrete experience Reflective observation Abstract conceptualization Active experimentation
<i>Transactive group memory</i>	Website content and hyperlinks (e.g.: FAQs, content) task-related archives and repositories (e.g.: the CVS) weekly digests archived discourse (e.g.: newsgroups archive, mailing list archive)	Productive inquiry Reflective observation
<i>Instructive content</i>	Online tutorials and screenshots Bug reporting system, CVS change log and diff application	Active experimentation Reflective observation Participative Practice
<i>Instructive discourse</i>	IRC (Internet relay chat)	Reflective observation Collective reflection
<i>Reflective discourse</i>	Asynchronous communication (e.g.: mailing lists, newsgroups)	Collective reflection Collective conceptualization Virtual experimentation ¹

Table 4. Learning Processes Initiated and Displayed Through Technological Tools
(Source: Hemetsberger and Reinhardt, 2006a, forthcoming)

Source: Hemetsberger and Reinhardt 2006a

Table 12: Learning Processes Initiated and Displayed Through Technological Tools

The FLOSSPOLLS Developer Survey revealed that except for other forms of self-study, which is performed by 58% of the community members, the most used ways to learn are those that provide the opportunity to either read or work on the code and that depend on Internet-based technologies (Table 13). Reading bug reports, participating in workshops or congresses, and learning by participating in training courses are the least used ways to learn.

Ways to learn	Percentage saying this is a very useful way to learn
Fix bugs	59,1
Other self-study	58,4
Reading source code of programs and patches	55,8
Reading other developers' feedback to my patches / bug-reports / bug-fixes	51,8
Checking programs' for errors and their causes	48,3
Reading books or articles on programming	46,0
Participate in the discussions within the FLOSS community	45,2
Reading bug reports	18,7
Participate in workshops or congresses	16,8
Learning by participating in training courses	11,7

n = 1453

Multiple responses ordered by shares (%) of community members who consider these ways to be useful

Source: FLOSSPOLLS Developer Survey 2005

Table 13: Ways to learn in the FLOSS community

A good example of how Internet-based technologies initiate and fuel learning processes within the FLOSS community is provided by the support system of FLOSS communities, which is as perfect as it is erroneous. On the one hand it offers a 24hours, 7 days a week, 365 days support with up to date content and learning materials, and all of this provided by volunteers at no charge. And on the other hand it is erroneous since none of this services are granted and consequently there might be less support at the individual level. Lakhani and von

Hippel (Lakhani & von Hippel 2002) analysed the support system of the Apache community and found out that their field support systems functions effectively and that 98% of the support services return direct learning benefits for the support provider. They are confirming that “giving as a natural thing” as described earlier by Demaziere, or “gaining reputation” and “personal enjoyment” are important motivational factors, but so are the learning benefits. Thus the situation between the information provider and receiver could be described as a win/win situation.

The fact that a great part of motivation to provide volunteering support resides in learning benefits for the support provider leads also to the conclusion that there need to be problems in order to keep the support system alive – or as Lakhani and von Hippel reported expressed this with a quote of Raymond:

“Actually . . . the list [of fetchmail beta-testers] is beginning to lose members from its high of close to 300 for an interesting reason. Several people have asked me to unsubscribe them because fetchmail is working so well for them that they no longer need to see the list traffic! Perhaps this is part of the normal life-cycle of a mature, bazaar-style project.” (Raymond, 1999, pp. 46–47).

Support in FLOSS is characterized by “information seekers posting their questions on a public website. Potential information providers log onto this website, read the questions and post answers if and as they choose to do so.” (Lakhani & von Hippel 2002) In FLOSS it is expected that information seeker first try to solve their problem themselves by the mean of available materials and if required by surfing the web.

For the case of Apache it might be interesting to note that the core development team made clear that they are not interested in providing any support and therefore the support system has evolved separately, operated by and for users themselves.

Lakhani and von Hippel also address the limits of FLOSS type support system regarding the number of support seeker in comparisons to support provider, and also the type of information requested. They found that about 2% of the knowledge providers were responsible for about 50% of the answers to questions posted on the help system and 50% of the questions were provided by 24% of the knowledge providers. The 100 most active information seekers posted an average of 10.43 questions and the 100 most active information providers posted an average of 83.63 answers during the 4-year period of their study. This suggests that only few individuals are active in providing answers to questions asked in the Apache system. Though it should be recalled at this point that mailing lists are rather frequented by developer, meanwhile the wider community might seek support at the forums.

Though providing a general overview on the technological perspective of FLOSS communities – in general as well as related to learning and knowledge management – it needs to be acknowledged that there is the need of providing further information on the role of wikis, on synchronous communication tools and about external environments such as myspace.

6. CONCLUSION: FLOSS LEARNING PRINCIPLES

In this chapter we summarise the main findings of each of the previous chapters and interpret them with regard to *principles of learning* in the FLOSS community. This chapter goes however beyond the scope of a pure conclusion as it also provides a first approach to the meaning these learning principles could have with regard to formal educational settings. This final chapter therefore also provides the ground for the next step of work in the FLOSSCom project, which is testing the transferability and applicability of these learning principles in formal educational settings. It should be noted however that this report does not anticipate the work of phase 2 (Summer University) of the FLOSSCom project, it only provides first rough ideas as a starting point for phase 2.

6.1. Openness and inclusivity

The examination of the demographics of the FLOSS community has illustrated that people access the community at any age (though at current it attracts more young people, which might be a reflection of the fact that the younger generation grew up with computers) and at any stage of professional career. This characteristic is usually known as “openness” or “inclusivity” of the FLOSS community.

It must however be noted that openness and inclusivity does not mean that there are no barriers to surmount. As the example of the exclusion of women shows, the FLOSS community, like any other social formation, has established specific cultural and social patterns and norms that require from anyone who wants to join a certain degree of assimilation. In the case of the FLOSS community, the specific cultural norms are obviously strongly determined by young males and establish a considerable hindrance for women. Openness and inclusivity does therefore only mean that those who want to join the community do not have to pass enrolment procedures or have to pass formal performance assessments.

With regard to formal educational settings we observe that the current overall age structure of the FLOSS community resembles very much the age structure of people in formal educational settings (students at school or university). The fact that professionals at an older age can join the community and learn illustrates that the FLOSS community provides an efficient means for lifelong learning. Provided that the FLOSS community will keep its capacity to sustain we assume that its age structure will change over the coming two decades and will finally

resemble the overall demographic structure.

6.2. Volunteering and volatility

The analysis of the structure of the community showed that members voluntarily decide which role(s) they want to play which responsibilities to take on. As a consequence, roles and responsibilities (or capacities) of community members can change over time and / or in different contexts. This results in a very vivid and volatile internal structure and dynamics of the community.

While volunteering and volatility is probably one of the cornerstones of the efficiency of the FLOSS community as a learning environment it is at the same time something that might be difficult to transfer to and apply in formal educational settings. While learning in the FLOSS community is efficient because “project managers” and “community managers” (and many more roles) voluntarily assume responsibility for organising work, tasks, content, and communication, in formal educational settings roles, tasks, and responsibilities are more pre-determined and rigid. While learning in FLOSS often results from a flow of marginal knowledge between community members and the role of “learner” and “teacher” can swiftly change with changing contexts, the roles of “learners” and “teachers” in formal educational settings are usually unequivocally attributed to persons, not to situations and contexts. Solutions for transferring volunteering and volatility to formal educational settings may foresee rewards for students who voluntarily assume positions similar to project or community managers in FLOSS. Alternatively, curricula could include the obligation of more experienced students to share their knowledge with the less experienced. For instance, computer scientists in the second year of education could be obliged to help first year students to solve a specific task. Obligation however contradicts the principle of volunteering, it will therefore be essential for the future work of the project to explore if and how such a replacement could work out. Finally, the range of roles that can be assumed in formal educational settings might be more limited than in the FLOSS community.

6.3. Using large-scale networks

Closely related to the abovementioned aspect of learning in FLOSS is the usage of large scale networks. Volunteering and volatility seem to be possible and effective because of the big size and continuous growth of the community.

The crucial question for transferring this principle to formal education is how similar networks can be created within formal environments, which usually have small classes. However, as the exploration of regular contacts between FLOSS community members has revealed, most FLOSS community members have regular contacts to only 1 to 5 other community members. It might therefore be possible to reap similar network effects from small networks in formal education as in the big FLOSS network.

6.4. Content-richness and specialisation

The analysis of motivations for participating in the FLOSS community and of the activities that are performed by its members revealed that the FLOSS community, though revolving on software development, offers a range of opportunities that by far exceeds the scope that is closely related to software. The double character of the FLOSS community as a community of practice, focussing on the techniques to produce software, and a social movement, focussing on strong social values such as freedom, appears to be very meaningful in this respect.

This content-richness of the FLOSS community however is not resulting in eclectic and disparate activities and outcomes. We found that the skills attainment in the FLOSS community focuses very much on the technical aspects, i.e. on skills needed for developing software and distributing it under an appropriate license. This seems to hold true especially for young newcomers. Moreover, FLOSS community members know that the skills they learn have a positive value on the labour market and are able to judge this value realistically. Precondition for being competitive to people with comparable formal degrees is that informally attained skills in the FLOSS community must be provable. Peer-reviewing is very important in this regard. Finally, surveys revealed that the FLOSS community provides skills that can specifically be better learned in this environment than in formal educational environments.

This might have a number of implications for formal education when FLOSS principles shall be implemented in this environment. First of all, formal education is usually very structured and targeted. To include aspects that are not directly related to what shall be taught and learned is considered to be ineffective. It must therefore be carefully explored if a FLOSS-like content-richness could (or should) be achieved in formal educational settings. If further research turns out that content-richness appears to be a precondition for a successful implementation of FLOSS learning principles in formal educational settings it must be

secured that content enrichment does not hinder professional advancement and specialisation of the students.

Regarding certificates, implementing FLOSS learning principles in formal educational settings might be feasible if (a part of) formal certificates and degrees are replaced by informal forms of recognising and acknowledging expertise. Certificates might have to become “weaker” and there might be a need for a bigger variety of “certificates” or proofs of participation. It might also be useful (or necessary) to replace pre-defined curricula that characterise many formal courses by a more flexible structure of learning, passing exams and getting degrees.

REFERENCES

- Abell, D., 1997. What makes a good case? *ECCHO: The House Journal of the European Case Clearing House*, **14** (Spring), pp.5-8.
- Aberdour, M., 2007. Achieving Quality in Open Source Software. *IEEE Software*, **24** (1), pp.58-64.
- Alzamil, Z. 2005. Towards an effective software engineering course project. *Proceedings of the 27th International Conference on Software Engineering, ICSE '05*. ACM Press, pp. 631-632.
- Ankolekar, A., Herbsleb, J.D. and Sycara, K., 2003. Addressing Challenges to Open Source Collaboration With the Semantic Web. In *Proceedings of Taking Stock of the Bazaar: The 3rd Workshop on Open Source Software Engineering, the 25th International Conference on Software Engineering (ICSE)*, Portland OR, USA, May 3-10 2003, pp.9-13. Available from: <http://www.cs.cmu.edu/~anupriya/papers.html> [cited 24 January 2007].
- Argyris, C, 1992. *On Organizational Learning*. Oxford: Blackwell.
- Bacon, Seb and Dillion, Teresa, 2006. *The Potential of Open Source Approaches for Education*. Futurelab, TeLearn Online, Available from: <http://telearn.noe-kaleidoscope.org/warehouse/bacon-2006-OpenSource.pdf>
- Barahona, J.M., Lopez, L. and Robles, G., 2004. Community structure of modules in the Apache project. *Proceedings of the 4th Workshop on Open Source Software Engineering*, Edinburgh, Scotland.
- Barahona, J.M., Tebb, C. and Dimitrova, V., 2005. Transferring Libre Software development Practices to the Production of Educational Resources: the Edukalibre Project. *First International Conference on Open Source Systems*, Genova, Italy, pp 341 - 348
- Barabasi, A. and Bonabeau, E., 2003. Scale-Free Networks. *Scientific America.*, **288** (5), pp.50-59.
- Barnett, C. and Ramirez, A., 1996. Fostering critical analysis and reflection through mathematics case discussions. In J.A. Colbert, K. Trumble, and P. Desberg, eds., *The Case for Education: Contemporary Approaches for Using Case Methods*. Boston, MA: Allyn and Bacon. pp.1-13
- Barrows, H.S., 1988. *The tutorial process*. Springfield: Southern Illinois University School of Medicine.
- Bechky, B.A., 2003. Sharing Meaning Across Occupational Communities: The Transformation of Understanding on a Production Floor. *Organization Science*, **14** (3), pp.312-330.
- Benham, M.K.P., 1996. The practitioner-scholars' view of school change: A case-based approach to teaching and learning. *Teaching and Teacher Education* , **12** (2), pp.119-135.
- Benkler, Y. (2002). Coase's Penguin, or, Linux and The Nature of the Firm. 29th TPRC Conference, 2001, arXiv:cs/0109077v2 [cs.CY]. Available from: <http://www.yale.edu/yalelj/112/BenklerWEB.pdf>
- Berends, B., 2002. Knowledge sharing and distributed cognition in industrial research, *Proceedings of the 3rd European Conference on Organizational Knowledge, Learning and Capabilities*, 5-6 April, Athens, Greece.
- Billet, S., 1994. Situating learning in the workplace-having another look at apprenticeships. *Industrial and Commercial Training*, **26** (11), pp.9-16.
- Billet, S., 2002. Critiquing workplace learning discourses: Participation and continuity at work. *Studies in the Education of Adults*, **34** (1), pp.56-68.

- Bliss, T. and Mazur, J., 1996. Creating a shared culture through cases and technology: The faceless landscape of reform. In J.A. Colbert, K. Trumble, and P. Desberg, eds., *The Case for Education: Contemporary Approaches for Using Case Methods*. Boston, MA: Allyn and Bacon. pp.15-28.
- Blumenfeld, P.C., Soloway, E., Marx, R.W., Krajcik, J.S., Guzdial, M. and Palinscar, A., 1991. Motivating project-based learning: Sustaining the doing, supporting the learning. *Educational Psychologist*, **26** (3/4), pp.369-398.
- Boud, D., and Feletti, G., 1997. *The challenge of problem-based learning*. 2nd ed. London: Kogan Page.
- Brown, J.S. and Duguid, P., 1991. Organizational Learning and Communities-of-Practice: Toward a Unified View of Working, Learning, and Innovation. *Organization Science, Special Issue: Organizational Learning: Papers in Honor of (and by) James G. March*, **2** (1), pp.40-57.
- Burnett, G., 2000. Information exchange in virtual communities: A typology. *Information Research*, **5** (4). Available from: <http://informationr.net/ir/5-4/paper82.html>
- Carrington, D. and Kim, S., 2003. Teaching Software Engineering Design with Open Source Software. *33rd ASEE/IEEE Frontiers in Education Conference*, Nov. 5-8, Boulder, CO. pp. S1C- 9-14 vol.3
- Chen, H., Shen, H., Xiong, J., Tan, S. and Cheng, X., 2006. Social Network Structure Behind the Mailing Lists: ICT-IIIS at TREC 2006 Expert Finding Track. *The Fifteenth Text Retrieval Conference (TREC 2006) Proceedings*, **12**
- Chen, H., Shen, H., Xiong, J., Tan, S. and Cheng, X., 2006. Social Network Structure Behind the Mailing Lists: ICT-IIIS at TREC 2006 Expert Finding Track. *The Fifteenth Text Retrieval Conference (TREC 2006) Proceedings*
- Chickering, A.W., 1976. Developmental change as a major outcome. In M.T.Keeton et al. eds., *Experiential learning: rationale, characteristics, and assessment*. San Francisco. pp.62-107.
- Coleman, J.S., 1976. Differences between experiential and classroom learning. In M.T.Keeton et al. eds., *Experiential learning: rationale, characteristics, and assessment*. San Francisco. pp.49-61.
- Coleman, J.S., 1995. Experiential learning and information assimilation. Toward an appropriate mix. In K. Warren, M. Sakofs and J.S. Hunt Jr., eds., *The Theory of Experiential Learning*. Boulder, Colorado. pp.123-129.
- Council of Europe, 2000. *Strategies for learning democratic citizenship*. Strasbourg, 19.7.2000.
- Crowston, K. and Howison, J., 2005. The social structure of Free and Open Source software development. *First Monday*, **10** (2),
- Crowston, K., and Howison, J., 2005. The social structure of Free and Open Source software development. *First Monday*, **10** (2). Available from http://firstmonday.org/issues/issue10_2/crowston/index.html [cited 18 Jan 2007].
- Cseh, M., Watkins, K.E. and Marsick, V.J, 2000. Informal and incidental learning in the workplace. In G.A. Straka, ed., *Conceptions of self-directed learning: theoretical and conceptual considerations*. Münster. Available from: <http://www.fsu.edu/~elps/ae/download/ade5385/Marsick.pdf>
- Dale, M. and Bell, J., 1999. Informal Learning in the Workplace. *DfEE Research Report 134*.
- David, P., 1998. Common Agency Contracting and the Emergence of 'Open Science' Institutions. *American Economic Review, Papers and Proceedings*. **88** (2), pp.15-21.
- David, P.A. and Foray, D., 2002. Economic fundamentals of the knowledge society. Available from <http://www-econ.stanford.edu/faculty/workp/swp02003.pdf>.

- Demaziere, D. (2006). How free software developers work, Môle Armoricaïn de Recherche sur la Soci t  de l'Information et les Usages d'INternet. 2007. Available from: http://opensource.mit.edu/papers/CLES_DDFHJ_juin_2006_vm_Anglais.pdf
- Demil, B. and Lecocq, X., 2003. Neither market nor hierarchy or network: the emerging bazaar governance. Available from <http://opensource.mit.edu/papers/demillecocq.pdf>
- Denning, P.J., 20024. Network laws, *Communications of the ACM, ACM Press*, **47**, pp.15-20.
- Desberg, P. and Fisher, F., 1996. Using technology in case methodology. In J.A. Colbert, K. Trumble, and P. Desberg, eds., *The Case for Education: Contemporary Approaches for Using Case Methods*. Boston, MA: Allyn and Bacon. pp.40-55.
- Dohmen, G., 2001. *Das informelle Lernen. Die internationale Erschlieung einer bisher vernachlssigten Grundform menschlichen Lernens fr das lebenslange Lernen aller*. Bonn, Bundesministerium fr Bildung und Forschung (BMBF).
- Dowdy, E., 2001. Birth pangs of a new learning environment: Collective representation, network technology, and communities of place. Paper Submitted for presentation at the Conference of International Communication Association, Washington, DC, US.
- Duch, B.J., Groh, S.E., and Allen, D.E., 2001. Why problem-based learning? A case study of institutional change in undergraduate education. In B. Duch, S. Groh, and D. Allen, eds., *The power of problem-based learning*. Sterling, VA: Stylus. pp.3-11.
- Duchastel, P., 2001. Learnability. Available from. <http://home.earthlink.net/~castelnet/info/learnability.htm> [cited 15 September 2001].
- Eraut, M., Alderton, J., Cole, G. and Senker, P., 2002. Learning from other people at work. In R. Harrison, F. Reeve, A. Hanson and J. Clarke, eds., *Supporting lifelong learning volume 1: Perspectives on learning*. London: Routledge. pp.127-145.
- Erdelez, S., 1999. Information encountering: It's more than just bumping into information. *American Association for Information Science*, **25** (3), pp.25-29.
- Esch, C., 1998. Project-Based and Problem-Based: The same or different? Available from <http://pblmm.k12.ca.us/PBLGuide/PBL&PBL.htm> [cited 11 September 2005].
- Faber, B.D., 2002. Educational models and open source: resisting the proprietary university. *Proceedings of the 20th Annual international Conference on Computer Documentation, SIGDOC '02*, ACM Press, pp.31-38.
- Faust, M. and Holm, R., 2001. Formalisierte Weiterbildung und informelles Lernen. In F. Becker, R.U.A Katerndahl, eds., *Berufliche Kompetenzentwicklung in formellen und informellen Strukturen. QUEM-report 69*, Berlin: ABWF. pp.67-108.
- Fisher, D., 2003. Social Networks for End Users, Survey Paper for Advancement to Candidacy. Available from: <http://www.bsos.umd.edu/gvpt/CITE-IT/Documents/Fisher%202003%20Soc%20Ntwks%20for%20End%20Users.pdf>
- Fitzgerald, B., 2004. A Critical Look at Open Source, *EEE Comp. Society*, **37** (7), pp.92-94.
- Garrick, J., 1998. *Informal learning in the workplace – unmasking human resources development*. London: Routledge.
- German, D.M. and Mockus, A., 2003. Automating the Measurement of Open Source Projects, *ICSE '03 Workshop on Open Source Software Engineering*. Portland; Oregon, May 3-10.
- German, M.D., 2005. Experience teaching a graduate course in Open Source Software Engineering. *Proceedings of the first International Conference on Open Source Systems*. Genova. pp.326-328.
- Ghosh, R.. and Glott (2005b), "The Open Source Community as an environment for skills development and employment generation". Proceedings of the European Academy of Management (EURAM) Conference, Munich, May 4-7. Also available online in a version

prepared for the European Commission: http://flosspols.org/deliverables/FLOSSPOLSD10-skills%20survey_interim_reportrevision-FINAL.pdf

- Ghosh, R.A., Glott, R., Krieger, B. and Robles, G., 2002. Free/Libre and Open Source Software: Survey and Study. Part IV: Survey of Developers. *Maastricht: International Institute of Infonomics / Merit*. Available from: <http://flosspols.org/deliverables/D16HTML/FLOSSPOLSD16-Gender Integrated Report of Findings.htm>
- Giuri, P., 2004. Skills and Openness of OSS Projects: Implications for Performance. Available from http://opensource.mit.edu/papers/giuri_etal.pdf
- Gloor, P.A., Laubacher, R., Dynes, S.B.C. and Zhao, Y., 2003. Visualization of Communication Patterns in Collaborative Innovation Networks - Analysis of Some W3C Working Groups. *CIKM '03: Proceedings of the twelfth international conference on Information and knowledge management*, pp.56-60.
- Gosh, R. and Glott, R. (2005a). FLOSSPOLs Skill Survey Report, Available from: <http://flossproject.org/papers/20050415/RishabGHOSH-padua-skills.pdf>
- Grupe, F.H. and Jay, K.K., 2000. Incremental cases: Real-life, real-time problem solving. *College Teaching*, **48** (4), pp.123-128.
- Habermas, J., 1981. *Theorie des kommunikativen Handelns*. Frankfurt a. Main: Suhrkamp.
- Hachen, D., 1996. Sociology cases database project. Available from <http://www.nd.edu/~dhachen/cases/> [cited January 16, 2001]
- Hakkarainen, K., Palonen, T., Paavola, S. and Lehtinen, E., 2004. Communities of networked expertise, professional and educational perspectives. *Advances in learning and instruction series*. London: Earli and Elsevier Ltd.
- Hanneman, R.A. and Riddle, M., 2005. *Introduction to social network methods*. Introduction to social network methods. Riverside, CA: University of California
- Hans, V., 2005. Some Myths of Software Engineering Education. *Proceedings of the 27th international Conference on Software Engineering, ICSE '05*. ACM Press, pp.621-622.
- Hazard, H., 1999. An action learning teacher reflects on case teaching *ECCHO: The House Journal of the European Case Clearing House*, **22** (Autumn/Fall), pp.5-8.
- Hazard, H., 2000. Action learning carried beyond case teaching. *ECCHO: The House Journal of the European Case Clearing House*, **24** (Spring), pp.5-8.
- Hemetsberger, A and Reinhardt, A.H.C.R. (2006a). "Learning and Knowledge-building in Open-source Communities - A Social-experiential Approach."
- Hemetsberger, A. and Reinhardt, C., 2004. Sharing and creating knowledge in open-source communities: the case of KDE. *Fifth European Conference on organizational knowledge, learning and capabilities*. Innsbruck, Austria.
- Hemetsberger, A., 2006. Understanding Consumers' Collective Action On The Internet: A Conceptualization And Empirical Investigation Of The Free- And Open-Source Movement. Available from <http://hemetsberger.cc/publications/pdf/habilitation.pdf>
- 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. Available from <http://opensource.mit.edu/papers/rp-hertelniednerherrmann.pdf> [cited 02 June 2004].
- Hippel, E. and Krogh, G., 2003. Open Source Software and the Private-Collective Innovation Model: Issues for Organization Science. *Organization Science*, **14**, pp.209-223.
- Hmelo-Silver, C.E., 2004. Problem-based learning: What and how do students learn? *Educational Psychology Review*, **16** (3), pp.235-266.

- Holmstrom, B., 1999. Managerial incentive problems: a dynamic perspective. *Review of Economic Studies*, **66**, pp.169–182.
- Houle, C.O., 1976. Deep traditions of experiential learning. In M.T Keeton et al., eds.,: *Experiential learning: rationale, characteristics, and assessment*. San Francisco.
- IEEE/ACM Joint Task Force on Computing Curricula, Software Engineering, 2004. Curriculum Guidelines for Undergraduate Degree Programs in Software Engineering. Available from <http://sites.computer.org/ccse/SE2004Volume.pdf> [cited November 27, 2005].
- Jarz, E.M., Kainz, G.A. and Walpoth, G., 1997. Multimedia-based case studies in education: Design, development, and evaluation of multimedia-based case studies. *Journal of Educational Multimedia and Hypermedia* , **6** (1), pp.23-46.
- Jensen, C. and Scacchi, W., 2007. Role Migration and Advancement Process in OSSD Projects. *International Conference on Software Engineering*
- Jonassen, D.H., 1992. Designing Hypertext for Learning. In E. Scanlon, and T.O'Shea., eds., *New Directions in Educational Technologies*. Berlin: Springer Verlag. pp.123-130.
- Jonassen, D.H., 1996. Scaffolding diagnostic reasoning in case based learning.
- Jonassen, D.H., 1999. Designing constructivist learning environments. In C.M. Reigeluth, ed, *Instructional design theories and models: A new paradigm of instructional technology*, Vol 2. Mahwah, NJ: Lawrence Erlbaum Associates.
- Jones, A., Scanlon, E., Tosunoglu, C., Morris, E., Ross, S., Butcher, P. and Greenberg, J., 1999. Contexts for evaluating educational software, *Interacting with Computers*, **11**, (5), pp.499-516.
- Julian, M.F., Larsen, V.A. and Kinzie, M.B., 1999. Compelling case experiences: Challenges for emerging instructional designers. *Performance Improvement Quarterly*, **13** (3), pp.164-201.
- Keeton, M.T. et al., eds., *Experiential learning: rationale, characteristics, and assessment*. San Francisco.
- Kilpatrick, W.H., 1921. Dangers and difficulties of the project method and how to overcome them: Introductory statement: Definition of terms. *Teachers College Record*, **22** (4), pp. 283-287 (ID Number: 3982). Available from <http://www.tcrecord.org> [cited 23 January 2006].
- Kinzie, M.B., Hrabe, M.E. and Larsen, V.A., 1998. Exploring professional practice through an instructional design team case competition. *Educational Technology Research & Development*, **46** (1), pp.53-71.
- Kirchhöfer, D., 2000. Informelles Lernen in alltäglichen Lebensführungen. Chance für berufliche Kompetenzentwicklung. Berlin. *QUEM-Report 66*.
- Koch, S. and Schneider, G., 2002. Effort, cooperation and coordination in an open source software project: Gnome, *Information Systems Journal*, **12** (1), pp.27-42.
- Kogut, B. and Metiu, A., 2001. Open-Source Software Development and Distributed Innovation. *Oxford Review of Economic Policy*, **17** (2), pp.248-264.
- Kolb, D. A. (1984). *Experiential learning: Experience as the source of learning and development*. NJ, Prentice Hall.
- Kollock, P. and Smith, M., 1997. *Communities and Cyberspace*. New York: Routledge.
- Krajcik, J.S., Blumenfeld, P.C., Marx, R.W. and Soloway, E., 1994. A collaborative model for helping middle grade science teachers learn project-based instruction. *The Elementary School Journal*, **94** (5), pp.483-497.
- Krishnamurthy, S., 2002. Cave or Community: An Empirical Examination of 100 Mature Open Source Projects. *First Monday*, **7** (2).
- Krogh G., Spaeth S, and Lakhani, K., 2003. Community, joining, and specialisation in open source software innovation: a case study, *Research Policy*, Vol.32. pp. 1217-1241.

- Krogh, G., Ichijo, K., and Nonaka, I., 2000. *Enabling Knowledge Creation: How to Unlock the Mystery of Tacit Knowledge and Release the Power of Innovation*. New York and Oxford: Oxford University Press.
- Krogh, G., Spaeth, S., Lakhani, K., 2001. Community, joining, and specialization in open source software innovation: a case study, *Research Policy*, **32** (7). pp.1217-1241.
- Krowne, A. (2005). The FUD-Based Encyclopedia, *FreeSoftwareMagazine*. 2007. Available from: http://en.wikipedia.org/wiki/Commons-based_peer_production
- Kuwabara, K. (2000). Linux: A Bazaar at the Edge of Chaos. *First Monday*, **5** (3).
- Lakhani, K, and Hippel, E., 2003. How open source software works: free user-to-user assistance, *Research Policy*, Vol.32, pp: 923-943.
- Lakhani, K.R., Wolf, R., 2002. BCG Hacker Survey. Available from <http://www.osdn.com/bcg>
- Lapachet, J.A.H., 1994. Virtual communities: The 90's mind altering drug or facilitator of human interaction. Available from http://is.gseis.ucla.edu/impact/s94/students/jaye/jaye_asis.html [cited September 20, 2003]
- Laudon, K., and Laudon, J., 2000. *Management Information Systems*, 6th ed. Upper Saddle River, N.J.: Prentice Hall International.
- Laurillard, D., 1987. Computers and the Emancipation of Students: Giving Control to the Learner. *Instructional Science*, **16**. pp.3-18.
- Lave, J., and Wenger, E., 1991. *Situated learning. Legitimate peripheral participation*. Cambridge: University of Cambridge Press.
- Lave, J., and Wenger, E., 2002. Legitimate peripheral participation in communities of practice. In R. Harrison, F. Reeve, A. Hanson and J. Clarke, eds., *Supporting lifelong learning vol. 1: Perspectives on learning*. London: Routledge. pp.111-126.
- Lehman, J.D, George, M., Buchanan, P. and Rush, M., 2006. Preparing Teachers to Use Problem-centered, Inquiry-based Science: Lessons from a Four-Year Professional Development Project. *The Interdisciplinary Journal of Problem-based Learning*, **1** (1), pp.76-99.
- Lerner, J. and Tirole, J., 2002. Some simple economics of open source. *Journal of Industrial Economics*, **50** (2), pp.197–234.
- Lin, Y. (2005a). Diversity of Knowledge and Dynamics of Knowledge Creation in FLOSS Communities, Vrije Universiteit Amsterdam. 2007. Available from: <http://www.feweb-vu.nl/dbfilestream.asp?id=2474>
- Lin, Y. (2005b). "Learn to solve problems: A virtual ethnographic case study of learning in a GNU/Linux Users Group." *The Electronic Journal for Virtual Organizations and Networks eJOV-Vol.7*: 56-69. Available from: http://www.ejov.org/Projects/264/Issues/eJOV%20Vol9/eJOV7_05_Huysman_Learn%20to%20solve%20problems%20a%20virtual%20ethnographic%20case%20study%20of%20learning%20in%20a%20gnu%20linux%20user%20group.pdf
- Lin, Y., 2005. Diversity of knowledge and dynamics of knowledge creation in FLOSS communities. Available at <http://www.feweb-vu.nl/dbfilestram.asp?id=2474>. [cited 20 March 2007].
- Liu, C., 2005. Enriching software engineering courses with service-learning projects and the open-source approach. *Proceedings of the 27th international Conference on Software Engineering, ICSE '05*. ACM Press. pp. 613-614.
- MacDonald, P.J., 1997. Selection of health problems for a problem based curriculum. In D. Boud and G. Feletti, eds., *The challenge of problem-based learning*, 2nd ed. London: Kogan Page. pp.93-102.

- Madey, G., Freeh, V. and Tynan, R., 2002. The open source software development phenomenon: An analysis based on social network theory. *Americas conference on Information Systems (AMCIS2002)*, pp.1806–1813.
- Maki-Komsi, S., Payry, P. and Ropo, E., 2005. Learning and knowledge building in distributed work environment. *The Electronic Journal for Virtual Organizations and Networks*, 7 (Dec.).
- Marsick, V.J., and Volpe, M., 1999. The nature and need for informal learning. In V.J. Marsick and M. Volpe, eds., *Informal learning on the job. Advances in developing human resources, no 3*. Baton Rouge: Academy of Human Resources Development. pp.1-9.
- Marsick, V.J., Volpe, M. and Watkins, K.E., 1999. Theory and practice of informal learning in the knowledge era. In V.J. Marsick and M. Volpe, eds., *Informal learning on the job. Advances in developing human resources, no 3*. Baton Rouge: Academy of Human Resources Development. pp.80-95.
- Maturana, H.R. and Varela, F.J., 1992. *The Tree of Knowledge: The Biological Roots of Human Understanding*. Boston, MA: New Science Press.
- Mayes, T., 1996. Why learning is not just another kind of work, *Proceedings on Usability and Educational Software design*, BCS HCI, London.
- McGivney, V., 1999. *Informal Learning in the Community. A trigger for change and development*. Leicester: NIACE
- Megias, D., Serra, J. and Macau, R., 2005. An International Master Programme in Free Software in the European Higher Education Space. *Proceedings of the first International Conference on Open Source Systems*. Genova, pp.349-352.
- Meiszner, A., 2007. Communication tools in FLOSS communities: a look at FLOSS communities at large, beyond the development team. Paper presented at the Web Based Communities Conference 2007, Salamanca, Spain.
- Michlmayr, M., 2004. Managing Volunteer Activity in Free Software Projects, *Proceedings of the 2004 USENIX Annual Technical Conference*, Freenix Track. pp.93-102.
- Mockus, A., Fielding, R. and Herbsleb, J.A., 2002. Two case studies of open source software development: Apache and Mozilla. *ACM Transactions on Software Engineering and Methodology*, 11 (3), pp.1-38.
- Moody, G., 2001. *Rebel code: inside Linux and the open source revolution*. New York: Perseus Press.
- Morine-Deshimer, G., 1996. What's in a case – and what comes out? In J. A. Colbert, K. Trumble and P. Desberg, eds., *The Case for Education: Contemporary Approaches for Using Case Methods*. Boston, MA: Allyn and Bacon. pp.99-123.
- Morison, T., 2001. Action Learning: A Handbook for Capacity Building Through Case-Based Learning. Available from: <http://www.adbi.org> [cited 11 December, 2001].
- Mutton, P., 2004. Inferring and Visualizing Social Networks on Internet Relay Chat. *iv, IEEE Computer Society*, 00, 35-43
- Nidumolu, S.R., Subramani, M. and Aldrich, A., 2001. Situated learning and the situated knowledge web: Exploring the ground beneath knowledge management. *Journal of Management Information Systems*, 18 (1), pp.115-151.
- Niedner, S., Hertel, G. and Hermann, S., 2000. Motivation in Open Source Projects: An Empirical Study Among Linux Developers. Available from: <http://www.psychologie.uni-kiel.de/linux-study>
- Nonaka, I. and Konno, N., 1998. The Concept of 'Ba': Building a Foundation for Knowledge Creation. *California Management Review*, 40 (3), pp.40-54.
- Nonaka, I. and Takeuchi, H., 1995. *The Knowledge-Creating Company*. New York: Oxford University Press.
- Ohira, M., Ohsugi, N., Ohoka, T. and Matsumoto, K., 2005. Accelerating cross-project

knowledge collaboration using collaborative filtering and social networks. *SIGSOFT Software Engineering Notes*, **30**, pp.1-5.

- OSS Workshop, Open Source Software Workshop, 2006. Open Source Software: Research Communities and Industries. Thessaloniki, 20 December, 2006. Available from: <http://www.sgo-oss.eu/events/public-workshop/>
- Overwien, B., 1999. Informelles Lernen: eine Herausforderung an die internationale Bildungsforschung. In P. Dehnbostel, W. Markert and H. Novak, eds., *Workshop Erfahrungslernen in der beruflichen Bildung – Beiträge zu einem kontroversen Konzept*. Neusäss. pp.295-314.
- Ozel, B., Cilingir, B. and Erkan, K. eds., 2006. Towards Open Source Software Adoption. *OSS 2006 tOSSad workshop proceedings*, Como, Italy. pp.79-88.
- Preece, J., Roger, Y., Sharp, H., Beyon, D., Holland, S. and Carey, I., 1994. *Human Computer Interaction*. Wokingham, UK: Addison Wesley.
- Raymond, E., 1999. The Cathedral and the Bazaar, *First Monday*, **3** (3),
- Raymond, E., 1999. *The Cathedral and the Bazaar: Musings on Linux and Open Source from an Accidental Revolutionary*. Sebastapol, CA: O'Reilly and Associates.
- Rheingold, H., 1993. The virtual community. Available from <http://www.rheingold.com/vc/book/2.html> [cited 6 July, 2003].
- Richardson, O., 2000. Developing and using a case study on the World Wide Web. *Journal of Educational Media*, **25** (2), pp.107-114.
- Savery, J.R. and Duffy, T.M., 1995. Problem-based learning: An instructional model and its constructivist framework. In B. Wilson, ed., *Constructivist learning environments: Case studies in instructional design*. Englewood Cliffs, NJ: Educational Technology Publications. pp.135-148.
- Savery, J.R., 2006. Overview of Problem-based Learning: Definitions and Distinctions. *The Interdisciplinary Journal of Problem-based Learning*, **1** (1), pp.9-20.
- Scacchi, W. (2005). Collaboration, Leadership, Control, and Conflict Negotiation in the Netbeans.org Open Source Software Development Community, Institute for Software Research, University of California, Irvine. 2007. Available from: <http://www.ics.uci.edu/%7Ewscacchi/Papers/New/Jensen-Scacchi-HICSS05.pdf>
- Scacchi, W. (2006). Free/Open Source Software Development: Recent Research Results and Methods, Institute for Software Research, University of California, Irvine. 2007. Available from: http://www.ics.uci.edu/%7Ewscacchi/Papers/New/Draft_Chapter_Scacchi.pdf
- Scacchi, W., 2001. Software Development Practices in Open Software Development Communities: A Comparative Case Study.
- Scacchi, W., 2002. Understanding the Requirements for Developing Open Source Software Systems.
- Scacchi, W., 2006. Understanding Free/Open Source Software Development Processes, *Software Process Improvement and Practice*, **11** (2), pp.95-105.
- Scharff, E. (2002). Applying Open Source Principles to Collaborative Learning Environments, University of Colorado, Center for LifeLong Learning and Design. 2007.
- Schon, D., 1987. *Educating the reflective practitioner: Toward a new design for Teaching and Learning in the Professions*. San Francisco: Jossey Bass.
- Schon, D., 1999. *The Reflective Practitioner - How Professionals Think in Action*. New York, NY: Basic Books.
- Senge, P.M., 1994. *The Fifth Discipline*. New York, NY: Currency Doubleday.
- Shockey, K. and Cabrera, P., 2005. Using open source to enhance learning. *ITHET 2005*. pp.7-12.

- Shulman, J.H., 2007. Case methods as a bridge between standards and classroom practice. Available from: <http://ericsp.org/pages/digests/shulman.pdf> [cited 11 December, 2001].
- Shulman, L.S., 1996. Just in case: Reflections on learning from experience. In J.A. Colbert, K. Trumble and P. Desberg, eds., *The Case for Education: Contemporary Approaches for Using Case Methods*. Boston, MA: Allyn and Bacon. pp.197-217.
- Silverman, R. and Wetly, W.M., 1996. Teaching without a net: Using cases in teacher education. In J.A. Colbert, K. Trumble and P. Desberg, eds., *The Case for Education: Contemporary Approaches for Using Case Methods*. Boston, MA: Allyn and Bacon. pp.159-171.
- Sowe, S.K, Angelis, L and Stamelos, I., 2006. Identifying Knowledge Brokers that Yield Software Engineering Knowledge in OSS Projects', *Information and Software Technology*, **48**, pp.1025-1033
- Sowe, S.K., 2007b. *An empirical study of knowledge sharing in free and open source software projects*, Ph.D. Department of Informatics, Aristotle University, Thessaloniki, Greece.
- Sowe, S.K., Karoulis, A. and Stamelos I., 2005. A constructivist view of knowledge management in open source virtual communities, In D.A. Figueiredo and A. Paula, eds., *Managing Learning in Virtual Settings: The Role of Context*, Idea Group, Inc, pp:290-308.
- Sowe, S.K., Karoulis, A., Stamelos I. and Bleris, G.L., 2004. Free-open source learning community and web-based technologies, *IEEE Learning Technology Newsletter*, **6** (1), pp.26-29.
- Sowe, S.K., Stamelos, I. and Deligiannis, I., 2006a. A Framework for Teaching Software Testing using F/OSS Methodology. In E. Damiani, B. Fitzgerald, W. Scacchi, M. Scott and G. Succi, eds., *IFIP International Federation for Information Processing, Open Source Systems*, **203**. Boston: Springer. pp. 261-266
- Sowe, S.K., Stamelos, I. and Samoladas, I., 2007. Emerging Free and Open Source Software Practices, *Idea Global*, May, 2007; p.4.
- Sowe, S.K., Stamelos, I., and Angelis, L., 2006b, An Empirical Approach to Evaluate Students Participation in Free/Open Source Software Projects. *IADIS International Conference on Cognition and Exploratory Learning in Digital Age CELDA 2006*. Barcelona, Spain. pp.304-308.
- Spinellis, D. and Szyperski, C., 2004. Guest Editors' Introduction: How Is Open Source Affecting Software Development?, *IEEE Software*, **21** (1), pp.28-33.
- Spinellis, D., 2003. *Code Reading: The Open Source Perspective, Vol. 1*, Pearson Education.
- Squires, D. and Preece, J., 1999. Predicting Quality in Educational Software: Evaluating for Learning, Usability, and the Synergy Between Them. Interacting with Computers, *The Interdisciplinary Journal of Human-Computer Interaction*, **11** (5), pp.467-483.
- Staring, K. (2005). Educational transformation through open source approaches, University of Oslo, Norway. 2007. Available from: <http://www.hia.no/iris28/Docs/IRIS2028-1106.pdf>
- Stinson, J.E., and Milter, R.G., 1996. Problem-based learning in business education: Curriculum design and implementation issues. In L. Wilkerson and W.H. Gijsselaers, eds., *Bringing problem-based learning to higher education: Theory and practice. New Directions For Teaching and Learning Series, No. 68*. San Francisco: Jossey-Bass. pp.32-42.
- Stürmer, M. (2005). Open Source Community Building, University of Bern, Switzerland. 2007. Available from: <http://opensource.mit.edu/papers/sturmer.pdf>
- Swap, W., Leonard, D., Shields, M. and Abrams, L., 2001. Using mentoring and storytelling to transfer knowledge in the workplace, *Journal of Management Information*

- Systems*, **18** (1), pp.95-114.
- Sykes, G. and Bird, T., 1992. Teacher education and the case idea. In G. Grant, ed., *Review of Research in Education, Volume 18*. Washington, DC: American Educational Research Association.
 - Thorn, B.K. and Connolly, T., 1987. Discretionary Data Bases: A Theory and Some Experimental Findings. *Communication Research*, **14**, pp.512–528.
 - Torp, L. and Sage, S., 2002. *Problems as possibilities: Problem-based learning for K-16 education*, 2nd ed. Alexandria, VA: Association for Supervision and Curriculum Development.
 - Tuomi, I. (2004). "Evolution of the Linux Credits file: Methodological challenges and reference data for Open Source research." *First Monday* 9(6). Available from: http://firstmonday.org/issues/issue9_6/tuomi/index.html
 - Valverde, S., Theraulaz, G., Gautrais, J., Fourcassie, V. and Sole, R.V., 2006. Self-Organization Patterns in Wasp and Open Source Communities, *IEEE Intelligent Systems*, **21**, pp.36-40.
 - Valverde, S., Theraulaz, G., Gautrais, J., Fourcassie, V. and Sole, R.V., 2006. Self-Organization Patterns in Wasp and Open Source Communities. *IEEE Intelligent Systems*, **21** (2)
 - von Hippel, E. and von Krogh, G., 2003. Open Source Software and the Private-Collective Innovation Model: Issues for Organization Science. *Organization Science*, **14** (2), pp.208-223.
 - von Krogh, G. et al., 2003. Community, Joining, And Specialization In Open Source Software Innovation: A Case Study, *Research Policy*, (32), pp.1217-1241.
 - Voß, G. and Pongratz, H., 1998. Der Arbeitskraftunternehmer. Eine neue Grundform der Ware Arbeitskraft? *Kölner Zeitschrift für Soziologie und Sozialpsychologie*, **50** (3), pp.131-158.
 - Ward, R., 1998. Active, collaborative, and case-based learning with computer-based scenarios. *Computer Education* , **30** (1/2), pp.103-110.
 - Watkins, K.E. and Marsick, V.J., 1992. Towards a theory of informal and incidental learning in organisations. *International Journal of Lifelong Education*, **11** (4), pp.287-300.
 - Wayner, P., 2000. *Free for All: How Linux and the Free Software Movement Undercut the High-Tech Titans*. New York: Harper Business.
 - Weiss, M. and Moroiu, G., 2007. In S.K. Sowe, I. Stanelos, and I. Samoladas, I., eds.,. Emerging Free and Open Source Software Practices Ecology and Dynamics of Open Source Communities. *IGI Global*, pp.48-67.
 - Wenger, E., 2000. *Communities of Practice and Social Learning Systems*.
 - Wenger, E.C., 1998. *Communities of Practice: Learning, Meaning, and Identity*. Cambridge: Cambridge University Press.
 - Wenger, E.C., 2000. Communities of Practice and Social Learning Systems. *Organization*, **7** (2), pp.225-246.
 - Wilkerson, L. and Gijsselaers, W., eds., 1996. Bringing problem-based learning to higher education: Theory and practice. *New Directions For Teaching and Learning Series, No. 68*. San Francisco: Jossey-Bass.
 - Williams, M.M., 1996. Using the case method in a foundations of education course. In J.A. Colbert, K. Trumble and P. Desberg, eds., *The Case for Education: Contemporary Approaches for Using Case Methods*. Boston, MA: Allyn and Bacon. pp.187-195.
 - Xu, J., Gao, Y., Christley, S. and Madey, S., 2005. A topological Analysis of the Open Source Software Development Community. *IEEE Proceedings of the 38th Hawaii International Conference on System Sciences, (IEEE, HICSS '05-Track 7)*

- Ye, Y., Yamamoto, Y. and Kishida, K., 2004. Dynamic Community: A New Conceptual Framework for Supporting Knowledge Collaboration in Software Development, *Asia Pacific Software Engineering Conference*. pp.472-481.