to be published:
Proc. of the 4th Workshop on Open Source Software Engineering at 26th International Conference on Software Engineering (ICSE04), Edinburgh, UK.

Inside an Open Source Software Community: Empirical Analysis on Individual and Group Level

Wolfgang Maass
=mcminstitute, University of St Gallen, CH-9000 St Gallen
wolfgang.maass@unisg.ch

Abstract

An established Open Source Software community (Apache Cocoon) was explored using an online questionnaire about demographic data and individual and group-related factors. Individual factors encompassed forms of contributions, motivation, expertise and knowledge. Role structures, expectations towards other members, trust and collaboration issues were analysed at group level. More than 60% of the developer community completed this questionnaire. Results provide a valuable basis for deeper understanding of knowledge sharing, collaboration and innovation processes in distributed work groups.

1. Introduction

Open Source software (OSS) communities have recently attracted attention for the impact they have on the global software market. This development especially enlivened research on the organisation and motivational structure of individuals (e.g. [1], [2], [3], [4], [5]).

OSS communities are virtual work groups consisting of members with skills in software development. They work in temporary, culturally diverse, geographically dispersed, electronically communicating work groups ([6]).

In contrast to commercial software vendors, members can freely join OSS communities and are unrestricted in their contributions. Virtual work groups have been investigated within given contexts ([7], [8]) but, compared to Linux or Apache communities, these groups have been rather small. Software development by virtual work groups is a long-standing topic, however if compared to larger OSS communities these studies also mainly focused on smaller projects [9]. In this sense, OSS communities provide unique opportunities for studying virtual work groups and distributed software development.

On one hand, the communication and collaboration of OSS communities are highly transparent. On the other hand, global distribution of community members and lack of job contracts with firms make it difficult to investigate communities on an individual level. Therefore most empirical studies on OSS communities concentrate on secondary logging information such as that provided by mailing lists, IRC chat logs and code repositories.

As a prerequisite for understanding how OSS communities communicate and collaborate, we need better knowledge of their beliefs, goals, attitudes, skill sets, communication and collaborative behaviour. Based on a detailed online questionnaire, the Apache Cocoon community was analysed at individual and group level. More than 60% of the developer community (34 valid responses) responded to this questionnaire.

2. Conceptual Foundations

Commercial software development rests on the ability to allocate technical and human resources in relatively complex organisational settings [10]. Labour is divided into specialist groups, such as technical programming teams, system engineering, integration/testing and project management [11].

Globally distributed software development by virtual teams promises the flexibility, responsiveness, lower costs, and improved resource utilisation necessary to meet ever-changing task requirements in highly turbulent and dynamic global business environments [12].

Open Source Software communities provide a role model for virtual teams that have inherited cultural, organisational and communicational schemes from the scientific community but then enriched and expanded the communication and collaboration processes step-by-step. From a psychological viewpoint, the perception of membership in some common social identity is constitutive for a group to exist [13]. Work groups are groups that are focused on the generation of problem solutions.

Any developer of OSS communities maintains a set of collective, social, norm-oriented and reward motives [2]. However, they also have the liability to deliver

contributions of a given quality for which rewards are granted in terms of gaining reputation within the community. Contributions depend on the expertise and skills that a developer obtains. Physical disconnection is regarded as being a negative factor for team performance [14]. OSS communities organically design organisational structures that intercept this threat and the known limitations of communication infrastructures.

Initial studies explored work group structures of OSS communities ([2], [3], [4], [1]). In a detailed study, Hertel et al. analysed motivational processes of members of the Linux kernel community. This study gave first insights into the demographic structure of parts of a community. Members were differentiated into two groups: developer group and interested reader group. On average, participants worked 18.4 hours per week on Linux and 20% of the developers received a salary for their Linux programming work. 38% were able to carry out Linux-related programming during their regular working hours, whereas the remaining 62% worked on Linux outside their regular work. The Linux kernel community is subdivided into teams ranging from two to 50 developers, with an average of 12. Studies found that the main motivational factors were identification factors, pragmatic motives, norm-oriented motives, social and political motives, hedonistic motives and motivational obstacles related to time losses. All factors were found to correlate positively with willingness to engage in the community, while lack of time is the biggest obstacle. Trust, in a limited meaning, only plays a minor role for motivation.

Established OSS communities such as Linux, Apache or mySQL emerged from small toy projects to fully-fledged software solutions for complex tasks. Accordingly, developers are required to provide mature expertise and development skills. Knowledge of members is codified [15] by information communicated to other members and most of all into documentation and source code. It depends on experience gathered over time and can be transformed into a shared experience that leads groups of people to encode, store, and retrieve relevant information together [16].

At group level, the organisation gives a formal structure to a community. OSS communities typically provide simple and clear-cut organisational structures.

Structural conditions for OSS organisations include a general culture, delegative and participative leadership principles, modular project structure, parallel release policy, motivating credit policy, and clear rules and norms [2]. OSS communities typically grow around an individual or a small team ([17], [18]).

The analysis of OSS communities in respect of their work group organisation is still an open field. Recent studies on open source software communities focus on the evaluation of indirect historical data, as provided by mailing list repositories and Concurrent Versions System (CVS) entries ([1], [19]). The question remains as to whether organisations of OSS communities provide a new model of business organisation [20].

Members in communities communicate and collaborate on the basis of mutual expectations, which in turn generate mutual trust or suspicion [21]. Several factors have been suggested to facilitate the development of trust: (1) shared social norms, (2) repeated interactions, (e.g., [22]), (3) anticipation of future association [23], and (4) physical proximity [24].

In respect of their communication, virtual work groups, and OSS communities in particular, are restricted in the exchanges considered to be relevant for building trust, warmth, attentiveness, and other interpersonal bonds within teams [25].

Software development can be described as a collaborative problem-solving activity where success is dependent upon knowledge acquisition, information sharing and integration, as well as the minimisation of communication breakdowns [3].

3. The Present Study

In order to get a broader empirical basis for research on virtual teams, we searched for an OSS community that possesses important attributes; such as whether it is well-established (at least 3 years), sufficiently large but not to large (50-100 developers), still in its growing phase, and with a vision on its software system that is relevant for business usage. Therefore we selected the Apache Cocoon community, which was established in 1998, includes some 40-60 active developers and has gained growing attention from firms for its architecture on XML publishing (cocoon.apache.org).

The questionnaire targeted (1) demographic data, (2) individual factors (contributions, motivation, expertise), and (3) group factors (organisation, mutual expectations, trust, collaboration, communication channels) as described in the previous paragraph.

4. Method

Data was gathered by an online questionnaire that was developed in cooperation with leading members of the Cocoon community in September 2003. The questionnaire was actively communicated by leaders at Cocoon's first get-together in October 2003, which was the reason why 42 responses (34 valid) were given, representing more than 60% of the developer community.

This questionnaire was designed to receive a broad understanding of the community on which further empirical studies can be designed. The response rate was surprisingly high even though the questionnaire contained 42 questions.

5. Results

5.1. Demographic information

The average age of Cocoon developers is 31.2 years (SD 6.17 years). 73.5% are from Europe, 20.6% from the USA and about 3% from Asia and Australia. Developers have become aware of Cocoon in three phases, showing characteristics of the diffusion function: 26.5% before 1999 (initialisation), 35.3% in 2000 (growth), 38.2% from 2001 to present date (stabilisation). On average, they started to contribute 1.6 years later (0.756, $p < 0.01$)[1].

On average, developers invest 3.1 hours per week in Cocoon (only 16% of the investment of Linux kernel developers [9]). Assuming a total number of developers of 50, this means a total investment of 150 hours per week, about 600 hours per month and 6,600 hours per year, which equals about 3.75 person years. According to the responses, 74% are paid by official jobs (Linux kernel only 38% [9]). 23.5% of Cocoon developers are self-employed, with a significant correlation to conducting business with Cocoon (.502, < 0.01). More than 60% of these people receive between 80-100% of their revenues from business with Cocoon.

5.2. Individual factors

The production fingerprint of an OSS community is given by its distribution of delivery types, such as patches, modules, problem reports, documentation, and ideas.

Members were asked which type of contribution they deliver frequently, periodically, rarely, or never. Only 6% report frequently and 20.6% offer patch contributions periodically, which is consistent with our analysis of CVS activities in Cocoon. The majority indicate that they rarely or never contribute patches. Modules are larger-scale conceptual and functional units consisting of a set of files. Almost 12% indicate that they contribute complete modules on a periodic basis. The correlation analysis shows an expectedly strong, positive relationship between patch and module contribution (0.501, < 0.01). As a rule, modules are provided by members who joined the community early.

The results show that 25% of all developers are active contributors of problem reports. 26.5% of all members write documentation frequently or periodically, which is an astonishingly high percentage. A significant relationship exists between people who write patches and documentation (0.39, < 0.05).

29.4% of members report frequent or periodic contribution of ideas, which significantly correlates with the activity of contribution of modules, documentation, and problem reports. Members who contribute ideas are those who are responsible for modules (0.709, < 0.01) than mere patches (0.364, < 0.01), which supports the view that ideas influence the overall architecture built on the basis of modules controlled and delivered by the same people.

Several significant correlations exist between contribution types, which indicates that active developers contribute anything from patches, modules, problem reports, ideas and even documents. People providing patches, modules and ideas tend to provide documentation to a lesser extent. Documentation is provided more by people who generate problem reports.

The motivation for participation in OSS communities is one of the key issues investigated by recent research [2]. Results from the Cocoon community support these findings showing that the main motivation is to learn new technologies (79.4%). 52.9% report that public utilisation is a main driver for their participation. Data shows a slightly negative correlation between personal branding and public utilisation, indicating a distinction between these two groups.

Teams of knowledge workers are inherently resistant to bureaucratic rationalisation and control [26]. Routine work and organisational issues dominate and are perceived to be a distraction from software engineering work. A hypothesis is that OSS communities are attractive to developers because routine work is omitted and the organisation is reduced to a minimum. This hypothesis can be supported by the result that 44.1% see self-realisation as a main driver for their participation. Further research is necessary for analysis of this hypothesis.

In summary, the analysis of motivational factors supports the view that people participate in the Apache Cocoon community because they want to learn new technologies from and with people with similar interests. Cocoon can be seen as a global family of skilled software developers. Results support the initial hypothesis that Cocoon is a global knowledge-sharing network.

Results show 65.7% of all members have a background in computer science and 24% in electro-engineering or general engineering. The questions regarding educational background show that 70.6% possess or are working on a university degree, while 14.7% possess a degree from an advanced technical college. 14.7% see themselves as self-educated, which represents those with a non-technical or at least a non-IT background.

Years of experience represent an important factor for professional programming. On an individual level it could be shown across various domains that it takes about 10

[1] based on Pearson correlations with a two-tailed significance test.

years to become a domain expert [27]. On average, Cocoon developers have 12.2 years of experience. Given the average age of 31.2, this means that Cocoon developers started programming at the age of 19. This in turn correlates with the beginning of university study.

Age and years of implementation correlate (0.787, < 0.01). Therefore, Cocoon can be separated into a cluster of programmers above 36 years of age and with more than 17 years of experience and a cluster ranging from 19 to about 31 years with 2 to 13 years of programming experience. It can be assumed that these two groups differ in their preferences regarding Cocoon. The correlation between age and the motivation to learn new skills (-0.339, < 0.05) and self-realisation (-0.414, < 0.005) is negative, which indicates that older Cocoon member are less interested in learning new skills or self-realisation than younger members.

5.3. Group factors

The study shows that 37% see themselves as "lurkers" who currently watch activities in the developer community and 71.4% see themselves as developers. The overlapping 8.4% of people watching and contributing can be interpreted as people who are at the edge of actual contribution. 34.3% hold the status of a committer, who dispose of the right to check in changes into CVS and 31.4% the status of a member of the program management committee (PMC). 93.6% of all committers are also members of the PMC.

On average, Cocoon members expect advanced expertise from other members. Members who report that they participate in order to learn new technologies (-0.394, < 0.05) and for self-reputation (-0.394, < 0.05) tend expect lower levels of expertise. Other correlations are rarely visible, so that as a new assumption we can assume that the expected level of expertise is an independent and basic variable.

As already indicated, not all members implement patches or modules but instead provide documentation, ideas, problem reports and others. Therefore we specifically asked for expectations of programming skills. Programming is considered the primary skill enabling an OSS community to deliver its primary product. This hypothesis is supported by the significant correlation between the expected level of general expertise and programming skills (0.874, < 0.01). In general, the expectation of programming skills is almost identical to that of general expertise. Hence, we can conclude that Cocoon members either seem to perceive their ability to program as their most important and dominant skill or any kind of contribution on an advanced level.

In general, 48.6% of all members perceive technical architecture as the most difficult lesson to learn. 20% find it hard to learn the rules of communication and 20% the rules of interaction. This supports the view that the biggest obstacle is also the biggest challenge because on one hand members have the main motivation to improve their technical skills and on the other they have a technical construct that challenges their skills.

28% of all members respond that they find it easiest to learn rules of communication or the "who's who". Together with the indication that people and their roles are also learned quite easily, it can be concluded that members learn the organisation and communication of the Cocoon community fast.

In respect of trust, on average members highly trust one another (average 2.6 on a scale from 1 [blind trust], 2 [very high], 3 [high], 4 [medium], 5 [low] and 6 [no trust]). Trust correlates with delivery of patches (0.415, < 0.05), delivery of modules (0.401, < 0.05), delivery of problem reports (0.380, < 0.05), delivery of documentation (0.411, < 0.05), and delivery of ideas (0.491, < 0.01). In contrast to users, Cocoon committers (0.514, < 0.01), PMC members (0.475, < 0.01), Apache Foundation members (0.364, < 0.05) are positively adjusted regarding trust towards other members. In summary, this presents a positive picture of trust relations between members. Users and lurkers show negative trust relations towards community members, developers tend to be positively biased, and from the status of a committer upward, members demonstrate very high trust in other members. Hence, in the Cocoon community trust seems to be derived from general work relationships rather than from physical interaction, which contrasts Handy's view that trust requires physical interaction (Handy 1995).

On average, Cocoon members claim to collaborate with 2.1 other Cocoon members, with a clear tendency to work with fewer people rather than with more. Committers (0.524, < 0.01), PMC members (0.566, < 0.01) and Apache Foundation members (0.560, < 0.01) significantly work with more members than regular developers or users (0.288). Significant correlations exist to developers who contribute modules (0.521, < 0.01), contribute documents (0.554, < 0.01), and contribute ideas (0.597, < 0.01). When correlated with the main contributions this pattern changes: significant correlations exist to discussions (0.467, < 0.01), votes (0.509, < 0.01), requirements (0.393, < 0.05). Also positive is the correlation with trust in other members (0.456, < 0.01) which indicates that the more people a member works with the higher is his or her trust into them. Furthermore, the more hours a member spends working for Cocoon the more people he interacts with (0.377, < 0.01). Finally it was found that the more revenue a member makes with Cocoon the more people he or she interacts with (0.512, < 0.01).

6. Implications

Work group analysis of OSS communities is an important but yet unexplored field. This study shows that mature OSS communities are able to attract high-profile experts. In contrast to functionally divided organisation of commercial software development projects, Cocoon developers account for almost any type of contribution, such as ideas, problem reports, patches, and modules. The motivation is largely determined by skill improvement and altruism (correlating with [2]). Developers expect other members to be highly skilled and experienced. They trust in other members to a large extent and perform their work in small teams or alone (correlates with "self-efficacy" [2]). This indicates that members perceive themselves as experts for a specific module, which will be valued by other members for its contribution to the overall system.

Various research questions can be derived from the findings of this empirical study. Further research is required on how individual knowledge is transmitted to other members based on limited communication channels and how it is transformed into transactive knowledge. These exchange processes depend on how members appraise each other and how this attitude evolves over time. Trust is a key concept for further analysis of related questions. Comparisons with centralised software development organisations will provide fruitful insights on performance differentials between centralised and virtual group organisations. From an economic point of view this will nurture research on globally distributed innovation and design processes.

7. References

[1] S. Koch and G. Schneider, "Effort, co-operation and co-ordination in an open source software project: GNOME", *Information Systems Journal*, vol. 12, pp. 27-42, 2002.

[2] G. Hertel, S. Niedner, and S. Herrmann, "Motivation of software developers in open source projects: An internet-based survey of contributors to the Linux kernel", *Research Policy*, vol. 32, pp. 1159-1177, 2003.

[3] G. von Krogh, S. Spaeth, and K. R. Lakhani, "Community, joining, and specialization in open source software innovation: a case study", *Research Policy*, vol. 32, pp. 1217-1241, 2003.

[4] L. Y. Moon and L. S. Sproull, "Essence of Distributed Work: The Case of the Linux Kernel", *First Monday*, vol. 5, 2000.

[5] E. von Hippel, "Innovation by user communities: learning from open-source software", *Sloan Management Review*, vol. 42, pp. 82-86, 2001.

[6] A. L. Kristof, H. P. Brown, and K. Sims Jr., "The virtual team: A case study and inductive model", in *Advances in Interdisciplinary Studies of Work Teams: Knowledge Work in Teams*, vol. 2, M. M. Beyerlein, D. A. Johnson, and S. T. Beyerlein, Eds. Greenwich, CT: JAI Press, 1995, pp. 229-253.

[7] M. K. Ahuja, D. F. Galleta, and K. M. Carley, "Individual Centrality and Performance in Virtual R&D Groups: An Empirical Study", *Management Science*, vol. 49, pp. 21-38, 2003.

[8] T. L. Griffith, J. E. Sawyer, and M. A. Neale, "Virtualness and Knowledge in Teams: Managing the Love Triangle of Organizations, Individuals, and Information Technology", *MIS Quarterly*, vol. 27, pp. 265-287, 2003.

[9] D. B. Walz, J. J. Elam, and B. Curtis, "Inside a software design team: knowledge acquisition, sharing, and integration", *Communications of the ACM*, vol. 36, pp. 62-77, 1993.

[10] R. W. Zmud, "Management of Large Software Development Efforts", *MIS Quarterly*, vol. 4, pp. 45-55, 1980.

[11] H. P. Andres, "A comparison of face-to-face and virtual software development teams", *Team Performance Management*, vol. 8, pp. 39-48, 2002.

[12] A. Mowshowitz, "Virtual organization", *Communications of the ACM*, vol. 40, pp. 30-37, 1997.

[13] J. C. Turner, "Towards a cognitive redefinition of the social group", in *Social identity and intergroup relations*, T. H., Ed. Cambridge, England: Cambridge University Press, 1982, pp. 138-161.

[14] C. Handy, "Trust and the virtual organization", *Harvard Business Review*, vol. 73, pp. 40-50, 1995.

[15] B. Kogut and U. Zander, "The Imitations and Transfer of New Technologies", *Organization Science*, vol. 3, pp. 383-397, 1990.

[16] D. Wegner, "Transactive memory: A contemporary analysis of the group mind", in *Theories of group behavior*, G. Mullen and G. Goethals, Eds. New York: Springer, 1986, pp. 185-208.

[17] A. Mockus, R. T. Fielding, and J. D. Herbsleb, "A Case Study of Open Source Software Development: The Apache Server", presented at Proc. of the 22nd International Conference on Software Engineering, Limerick Ireland, 2000.

[18] C. Robottom Reis, "An Overview of the Software Engineering Process and Tools in the Mozilla Project", presented at Workshop on Open Source Software Development, Newcastle upon Tyne, 2002.

[19] G. von Krogh and E. von Hippel, "Special issue on open source software development", *Research Policy*, vol. 32, pp. 1149-1157, 2003.

[20] J. Lerner and J. Tirole, "The Simple Economics of Open Source", National Bureau of Economic Research, Cambridge, Working Reports 7600, 2000.

[21] M. Deutsch, "Trust and Suspicion", *The Journal of Conflict Resolution*, vol. 2, pp. 265-279, 1958.

[22] J. L. Bradach and R. G. Eccles, "Markets versus hierarchies: From ideal types to plural forms", *Annual Review of Sociology*, vol. 15, pp. 97-118, 1989.

[23] W. W. Powell, "Neither market nor hierarchy: Network forms of organization", *Research in Organizational Behavior*, vol. 12, pp. 295-336, 1990.

[24] B. Latane, J. H. Liu, M. Nowak, M. Benovento, and L. Zheng, "Distance matters: Physical space and social impact", *Personality and Social Psychology Bulletin*, vol. 21, pp. 795-805, 1995.

[25] S. L. Jarvenpaa and D. E. Leidner, "Communication and Trust in Global Virtual Teams", *Organization Science*, vol. 10, pp. 791-815, 1999.

[26] M. Reed, "Organizational theorizing: a historically contested terrain", in *Handbook of Organization Studies*, S. R. Clegg, C. Hardy, and W. R. Nord, Eds. London: Sage, 1996.

[27] K. A. Ericsson, R. T. Krampe, and C. Tesch-Römer, "The role of deliberate practice in the acquisition of expert performance", *Psychological Review*, vol. 100, pp. 363-406, 1993.