Science, Technology & Innovation Studies
Vol. 4, No. 1, July 2008

ISSN: 1861-3675

STI Studies

www.sti-studies.de

# Balancing Requirements of Decision and Action: Decision-Making and Implementation in Free/Open Source Software Projects*

**Niels C. Taubert** (University of Bielefeld)

received 30 June 2006, received in revised form 16 June 2008, accepted 26 June 2008

## Abstract

This article deals with decision-making processes about new development aims in Free/Open Source software (FOSS) projects. It focuses on the question how community driven projects manage to not only make decisions but also implement them successfully. Following the approach of Nils Brunsson, the requirements of (rational) decision-making and action are somewhat antagonistic: On the one hand, rationality of decision-making implies extensive evaluation of alternatives and arguments that can lead to an uncertainty as to which of the alternative will be chosen. On the other hand, a good basis for collective action is established when uncertainty is reduced and consistent expectations exist as to what kind of action will be performed. Corroborating on an empirical analysis of a decision-making process and interviews conducted with FOSS developers, three mechanisms of ending a discussion are identified. The paper concludes evaluating to what extent each of these mechanisms serves the requirements for decision-making and action.

* The research for this paper has been carried out in a PhD project within the Research Training Group 'Genese, Strukturen und Folgen von Wissenschaft und Technik' at the 'Institute of Science and Technology Studies' (Bielefeld University). The paper is largely indebted to the discussion of an earlier version of the paper in the 'Methodenwerkstatt' held by Alfons Bora and to Alfons Bora's advice. I would like to thank the DFG for supporting the PhD-project with a doctoral grant and I would also like to thank two anonymous referees of STI-Studies for their helpful and instructive comments. Without the help of Felicitas Krämer and Marc Weingart I would not have been able to write this article.

## 1 Introduction

The field of Free/Open Source Software (FOSS) development has become a field of interest for the disciplines concerned with technological development and innovation activities. Although the phenomenon is still quite new there are a variety of aspects that have already been studied in detail. In this paper I will focus on a particular aspect of FOSS development. I will deal with the question, how a certain type of FOSS project – so called community driven projects – manage to decide about their aims and to implement these decisions. In these projects, developers mainly participate on a voluntary basis, and most projects display a low degree of role differentiation and a weak hierarchy. Thus, it is neither self-evident nor trivial that these projects manage these tasks. But the existence of FOSS shows that there must be a solution to the decision-making and implementation problems. This observation serves as a starting point. By analyzing a decision-making process I will show that community driven FOSS projects are not only orientated to decision rationality but also have to consider how to implement the decisions through collective action. However, the analysis undertaken in this paper has a limited scope in two respects. Firstly, it deals with decision-making and implementation only with respect to a certain type of community driven FOSS projects. In other kinds of FOSS projects different mechanisms may exist. Secondly, the paper has an explorative nature. It analyzes decision-making in FOSS projects on an empirical ground, but is far from providing final evidence that there are not more mechanisms or even that the mechanisms analyzed here are the most important ones.

The paper is organized as follows: In providing an overview of the literature of the FOSS development, I will highlight some main characteristics of FOSS projects that are important for the question of collective action. Subsequently, a theoretical framework is developed that allows to differentiate between decision-making on the one hand, and actions to implement these decisions on the other hand. In the fourth section the methodology of the analysis is outlined and the case of a community driven project is introduced. The fifth section deals with a decision-making process and its implementation. The various factors that influence this process are analyzed and enriched with findings derived from interviews with FOSS developers. In the conclusion the results will be summarized, and it will be evaluated how well different outcomes of decision-making processes meet the requirements of rational decision-making and action.

## 2 FOSS: Main characteristics[1]

The distinction between Free/Open Source Software and other types of Software is based on different types of licenses by which the use of the software is regulated (Stallman 2002: 41). FOSS is protected through a special license permitting everyone unrestricted use, copying, distribution, and modification. Other software licenses that do not grant these four rights to everyone make software proprietary.[2] The first, most important and widely used license[3] guaranteeing permissive

[1] We can only refer to a selected body of research on Free/Open Source Software development here. For a very good overview of the current state of the discussion cf. von Krogh/von Hippel (2006: 976-982) for management science and Holtgrewe/Brand (2007: 28-30) for the social sciences more generally.

[2] There are of course various proprietary software licenses for different purposes. For the argumentation developed here, the rough distinction between free and non-free software is sufficient. For a category of free and non-free software see the website of the GNU-project: (http://www.gnu.org/philosophy/categories.html, last access 03/2008).

[3] Lerner and Tirole 2002a; Bonaccorsi/Rossi 2003a: 9, 2003b: 1248. O'Mahony (2003) argue that FOSS licenses

application of software is the GNU General Public License (GNU GPL).[4] In contrast to some of the FOSS licenses,[5] it specifies one important restriction: the terms of the license have to be applied to any modified or non-modified version of the program. As a consequence, permissive application of the program and its derivates is guaranteed in the future. This 'repetition clause' makes a program 'copyleft' and prevents FOSS from being changed into proprietary software.[6] The consequence of the license is that FOSS becomes a privately produced public good (O'Mahony 2003: 1180) without rivalry in consumption. [7]

FOSS can be developed in various ways. It is not uncommon that a FOSS program is developed by an individual programmer,[8] produced behind closed doors of a firm (and released) and distributed as FOSS after completion, or is produced in projects in which individual developers cooperate with firms.[9] But there is also a unique social structure that can only be found in the field of FOSS: The original and still very important – in terms of numbers of software projects – type is 'community founded' and predominantly 'community driven'. In this type of project, the social structure in terms like e.g. the pattern of decision-making and the coordination of programming activities arises by self-organisation. This structure first appeared in 1991 in the Linux project, developing a free operating system for different hardware platforms.[10] This kind of social structure has a special feature: It is often highlighted that the absence of any technical restrictions and free access to the project infrastructure enables anyone who is interested to participate. In principle, each participant can pose questions, suggest new aims of the project, monitor and participate in decision-making processes, and even contribute to the code of the program.

One important focus of research on FOSS projects concerns the motivation of the participants. It was Lerner and Tirole (2000, 2002b) who asked, taking a rational-choice perspective, why people should contribute to the production of a common good if no one can be excluded from its use even when not having contributed to the production of the good. Lerner and Tirole argue that there are various factors guiding developers to participate in FOSS projects. They suggest to distinguish between immediate benefits like payment, fixing a bug, or customizing the program to one's own needs

---

are an integral part of a more complex system of regulations that include legal and normative sanctions, incorporation of the project, individual copyright transfer and a protection of trademark brands.

[4] The GNU project was the first to develop free software. For the history of the project, see Stallman (2002: 15-30; 1999). The aim of the GNU GPL is to protect the work of the project and prevent FOSS from being turned into proprietary software. For the original terms of the license see: (http://www.gnu.org/licenses/gpl.html, last access 04/2008).

[5] An example for this type is the BSD License that, like all other free software licenses, allows free use, copy, distribution and modification. For the original terms of the license, see: (http://www.opensource.org/licenses/bsd-license.php, last access 06/2006).

[6] The GNU GPL also has an 'infective character'. If some code protected by the GNU GPL is used in a larger work the license enforces that the GNU GPL is applied for the whole program (see Holtgrewe/Werle 2001: 54).

[7] For this reason, the frequently used term 'almende' (e.g. Grassmuck 2002) is somewhat misleading. An important characteristic of the almende is that there is rivalry of consumption. This characteristic leads to an overexploitation of the resource and a 'tragedy of the commons' (Hardin 1968).

[8] Ghosh/Robles/Glott (2002: 19) found out that the vast majority of the FOSS projects is carried out by one or two developers.

[9] Like RedHat, SUSE and Mandriva Conectiva for example.

[10] Eric Raymond highlights the relevance of the innovation of open software projects in his influential essay '*The Cathedral and the Bazaar*' (1999: 27-78). For a sociological analysis of this organisational innovation and the consequences for the development of FOSS see Taubert 2006: 72-87.

on the one hand and delayed benefits and rewards on the other. To the latter count ego gratification incentives (peer recognition) and career concern incentives that may lead to future monetary rewards (Lerner/Tirole 2002b: 213 f.).

In contrast to these early explanations of participations in FOSS development, other scholars highlight that – especially in the case of community driven projects – the intrinsic interest in developing software itself is one important incentive (cf. Osterloh/Rota/Kuster 2002; Hertel/Niedner/Herrmann 2003; Lakhani/Wolf 2005; Taubert 2006). Others claim that it is even the most important factor (Brand/Holtgrewe 2004: 17) that leads to contributions in FOSS projects. Futhermore, surveys with FOSS developers show that they feel highly creative while tackling development problems, and they frequently or always lose track of time (Lakani/Wolf 2005).

A critique of the earlier rational-choice explanation of Lerner and Tirole concerns the assumption that a situation of choices precedes the contributions. It is suggested to differentiate between the first contribution and enduring engagement in FOSS projects. Exhaustive consideration of costs and benefits are more likely to occur in the first than in the second case (Taubert 2006: 141). Other studies show that the motives for a first participation differ from the motives of a long-term participation (Shah 2006: 1004), and that the relevance of intrinsic motivations increases in long-term participation. Moreover, it is supposable that the relevance of different factors varies with the type of the project. In the case of community driven projects it seems plausible to assume that monetary payment plays a less important role than in projects where software companies contribute.

Another research focus on FOSS projects concerns the way the development process is organized. A common observation is that the degree of involvement varies to a large extent and that the group of highly involved developers is relatively small. For example Koch and Schneider (2002) found out that the majority of the 301 developers identified contribute a small amount to the code[11] while the 15 most active developers contribute 48% of the lines of the code (Koch/Schneider 2002: 30). Those findings suggest that these highly active developers also make the decisions. Thus, the group of decision makers is relatively small. Although no one in particular is the owner of a certain program, a certain role structure seems to exist. In the literature it is very common to distinguish between the maintainer, the core-developer and the community of users (Grassmuck 2002: 237-239). The different groups of participants not only vary with respect to the degree of activity, but also with regard to their contributions. For instance, *users* report bugs and sometimes also suggest solutions. *Co-developers* participate in these activities but also analyze and contribute to the code. *Core-developers* and the *maintainer*, however, contribute to the already mentioned activities, but are additionally involved in decision-making processes (Gläser 2006: 270).

But even if the number of participants involved in decision-making is small, how are decisions made in FOSS projects? Some authors highlight the role of 'leadership' or 'leadership-teams' and the moral authority of the maintainer (e.g. Lerner/Tirole 2001: 823). This is indeed true for some of the prominent projects of high strategic importance, such as Linux and Apache (Lerner/Tirole 2002b). But there are also big projects in which the degree of formalization of the organization is low.[12] In these projects the question of how the participants manage to decide

[11] In terms of lines of the code.

[12] An example for this type of project is the K Desktop Environment (KDE) that develops a graphic user interface for Unix- and Linux Operation Systems (See Holtgrewe/Brand 2007: 36).

about and pursue their aims without having recourse to a hierarchical modus of decision-making seems to remain open. In the next section, a theoretical framework is developed, which is adequate to explain for decision-making processes in the case of FOSS projects.

## 3 Theoretical Framework

In sociology and management science a large body of literature about decision-making exists. An overview of the most important concepts could start with the rationalistic tradition rooted in theory of bureaucratisation by Max Weber (1972). It postulates that organizations are rational actors that make decisions on the principles of impersonal application of rules, records and control. Important contributions that should be mentioned are the critique of the assumption of rationality of decision-making in Cyert/March (1963) and March (1994), and the garbage can model of decision-making where decision rationality seems to be lost (Cohen/March/Olson 1972). It is not the place here to unfold such an overview. One thing I wish to highlight by mentioning the work of these eminent scholars of the field is the focus of these theories of decision-making. The focus is very much on the 'logic of the choice' of alternatives and far less concerned with processes of the implementation of the decisions.[13]

A more adequate theory about decision-making in FOSS projects should offer a broader perspective. The aim of FOSS projects is not to make decisions but to develop software. Therefore, making a choice is not an end in itself but a step towards the implementation of a decision. A scholar in the field of organisational studies, who offers a theoretical framework that allows the integration of both aspects, is Nils Brunsson. In the first of his main works, 'The Irrational Organization' (1985), he starts with an overview of the main components of classical management theory and its normative decision-making theory. The picture drawn there suggests that managers mainly deal with decision-making and ample suggestions are made as to how the rationality of decision-making can be improved. In this context 'rationality' means that managers make decisions on the ground of stable preferences, careful consideration of all alternatives regarding the costs and benefits and the likeliness that these costs will occur and the benefits will be realized.[14]

From this starting point Brunsson comes to the common observation that decision-making in real life organizations frequently violates the rules of rationality. He does not intend to explain the differences between the normative standards of rational decision-making and empirical decision-making processes in a 'chauvinist' manner. Examples for those explanations would be that "subjects are not clever enough to behave rationally"[15] (Brunsson 1985: 17), that there are "certain types of irrationality" "inherent in the human character" (ibd.) or that there are "practical constraints" (ibd.). Brunsson, however, does not argue that these explanations are fundamentally wrong. Instead, he formulates a critique on the way the topic 'decision-making' is usually framed: „A decision making perspective fails to recognize that managers do more than make decisions. Making a decision is merely a step toward action. The decision is not the end product. Managers get things

[13] This observation has been made by Brunsson (1985): aside from his contribution the situation did not change much.

[14] In this field the rational-choice paradigm is very prominent. For an overview see Elster (1986).

[15] This kind of assumption can also be found in the concept of 'bounded rationality'. In this concept it is argued that, compared to the complexity of the world, the capacity of the human mind for formulating and solving problems is low (Simon 1957: 198; March/Simon 1958, March 1978).

done – act and induce others to act." (ibd. 18) Therefore, Brunsson suggests extending the perspective from the narrow focus on decision-making to a broader perspective: on decision-making and its implementation in organisational action.[16] With this change of the perspective a good deal of deviation from 'rationality' can be explained through the demands of 'action'. To put it differently, many aspects of decision-making that seem to be irrational from a decision-making perspective can be regarded as rational from the action point of view, if they improve the conditions for collective action.

What are the requirements of action? And what aspects does a decision maker have to take into account in order to prompt action? Brunsson points out that, for organisational action, different actors have to cooperate, and that a common cognitive, motivational, and commitment-related ground has to be reached. First, in the cognitive dimension it is important that there are consistent expectations about future action. Members of the organisation find it worthwhile to act only if they believe that "their doing will result in an organizational action" (Brunsson 1985: 19). If the individuals are not sure whether or not an organisational action is going to take place, they will not find it worthwhile to act. The second condition for organisational action is motivation. In this concept, 'motivation' means that people desire to contribute to the organisational action with their individual action. They will merely contribute if they regard the aim of the organisational action as a good thing (ibd. 19). The social aspect of action is commitment. This third condition for organisational action can be described by the fact that the members of the organisation trust on a certain type of behaviour, or attitude, which is shared by the rest of the organisational members involved in the action. If they do not trust in the existence of this attitude, or behaviour, they are not willing to take part in the action (ibd. 20). At this point of Brunsson's argumentation it appears to be clear that a certain behaviour, which leads to an improvement of decision rationality (e.g. taking more alternatives into account, analysing the consequences of an action in greater detail and so on), does not necessarily improve the conditions for organisational action.

In community driven FOSS projects the requirements of actions deserve closer attention because of specific framework conditions under which decision-making and the implementation of decisions takes place. Like other organisations solely relying on voluntary (unpaid) work, the projects themselves usually do not have financial resources that could be used for motivational purposes. This feature has an effect on the creation of the conditions for action: Since a lack of agreement cannot be compensated by financial means, the motivation to participate in collective action depends to a large degree on considering the chosen action as a 'good thing'. Therefore, one can expect that the agreement on a specific aim be of higher relevance in the case of community driven projects than in organisations, which can offer other resources for the motivation of the members.

In order to better understand the requirements of action, Brunsson gives a variety of practical examples of techniques for the improvement of the conditions for collective action. The main scope of these techniques is to reduce uncertainty, since uncertainty obstructs the cognitive, motivational and commitment-related conditions for action. Here, I will stress only two of

[16] Brunsson's theory only deals with a specific type of action: organisational action for change. Action means any activity that is not purely cognitive in character, organisational action means that action is „accomplished by several organization members in collaboration" (Brunsson 1985: 6), and action for change means „that a new kind of organizational action is undertaken, or that a previous type of action is discontinued, or both" (ibd. 9).

them. A first technique or strategy is to limit the numbers of alternatives taken into account. This helps to reduce the degree of uncertainty and makes it more likely that a given action is going to take place. A typical way to limit the alternatives is to propose alternatives that are clearly unacceptable, in order to highlight the advantages of the one (and only) acceptable alternative (ibd. 23). From a decision point of view this behaviour is irrational. But from an action point of view this strategy is rational: in the motivational dimension this strategy clarifies which alternative is desirable and in the cognitive dimension it helps to elucidate the expectations about which option will be chosen, and what kind of organisational action will be performed.

Brunsson describes a similar technique, which concerns the assessment of consequences. The rational calculation of the likeliness of positive and negative outcomes and the exhaustive assessment of the consequences is highly rational from a decision-making point of view. But as far as it creates uncertainty, it is highly irrational from an action point of view. A technique to avoid uncertainty is to reduce the rationality of decision-making by looking at the consequences in one direction only, by assessing desirable consequences for the acceptable alternative, and by suppressing any negative consequences it might have (ibd. 28). This strategy aims to improve the conditions for action at least in the motivational dimension.

But these conditions also depend on the outcome of collective decision-making. It makes a difference if the process of decision-making is concluded through consensual agreement, compromise or disagreement.

- The most 'harmonic' outcome of a decision-making is consensus. All actors involved in the decision-making process are convinced that the chosen alternative is the appropriate one, and there is no antagonism within the participants of the project. The absence of antagonism is not necessarily the result of a rational and extensive discourse, but can emerge in different ways. For example, it can arise in situations where only a few alternatives are taken into consideration and it is obvious which one is preferable.
- Compromise is a distinct outcome of a decision-making process. Although all actors accept the outcome, the compromise is in no accordance with the interest and preferences of at least one actor. Actors usually agree to a compromise after bargaining on the ground of the insight that it is the best result that can be reached if the diversity of perspectives, interests, and preferences of the other actors involved are taken into account.
- A third outcome is dissent. Here, neither consensus nor compromise can be reached in a decision-making process, and the antagonism still persists. In the case of FOSS-projects, and with respect to the action dimension, different situations can arise: (a) the development process stagnates, (b) the project splits up in different sub-projects (forking), or (c) an alternative is enforced by an actor, (and is accepted by the others).

Before starting to analyze a decision-making process in a community driven FOSS-project – and before considering the question of how the specific decision outcomes and routines serve the requirements of decision-making and action – the methodology on which the data collection is based will be outlined.

## 4 Methodology

The following analysis is based on a case study of a project that serves as a typical example of a community driven project. The project was selected because it met the following criteria:

- *Size of the project*: The number of developers involved in FOSS projects vary on a large scale between the many one- or two-person-projects and the few big (and mostly very famous) projects where some hundred developers are involved. Therefore, it was ensured that the selected case would be big enough that problems of coordination would most probably appear. The big and famous projects were excluded because they represent extreme cases, with partly exceptional social structures and coordination routines.
- *Duration of the project/success*: To find a case where established solutions or mechanisms for coordination and decision-making can be observed, a project was selected that has already released a stable version[17] of the program
- *Mailing list with an archive*: This criterion was set up for practical reason. It was formulated to guarantee easy access to the earlier communication of the project.
- *Type of the program*: Due to an interest in the influence of users for the development of software in the study, (although this is not the main focus here) a project was selected that develops a program, which addresses users who do not have to have competencies in programming. On account of this criterion all projects developing programming tools were excluded.

One project that meets these criteria is 'KMail', which develops an email client for the desktop environment 'KDE'. It includes functions such as sending/receiving emails, tools for writing emails (editor and spell checker), an address book, and the integration of PGP encryption. From a user perspective, it resembles other email clients such as Microsoft Outlook or Mozilla/Netscape Mail & Newsgroup. Since the foundation of the project in 1997, 48 project members have worked intensively on the project and made substantial contributions to the program. As a result, they are listed as 'authors' on the project's website.[18] Prima facie and with respect to the number of developers involved, the project seems to be a large one. However, this impression needs to be put into perspective by considering the high level of fluctuation: Most developers join the project, work on a part of it for a while, and then leave after the work on this specific part is done. Only the project's maintainer and the core developer remain involved for longer periods. In the case of KMail, usually less than ten (core-) developers work on the project simultaneously.

The KMail project is based on an advanced technological infrastructure. Its website provides information about the program, its features, and its authors;[19] a bug-track system for collecting user feedback on the program's unexpected behavior;[20] a download site where one can obtain the latest versions of the program; and a current version repository (CVS) for the management of the development version of the source code on which the members of the project are working. Communication concerning the development of the program takes place between the programmers on a mailing list. This list is

[17] A feature of FOSS is that new versions of the program are published rapidly. The projects distinguish between developer versions or unstable versions on the one hand (that are used by developers in order to remove bugs and make the program more reliable), and official releases on the other hand (that are well proven and that are intended to be used by users, who do not have any special technical competency in programming for their normal day-by-day use).

[18] See: (http://kontact.kde.org/kmail/authors.php, last access 03/2008).

[19] See: (http://kontact.kde.org/kmail/, last access 03/2008)

[20] KMail shares a bugtracking system with the master project KDE of which KMail is part of. See: (http://bugs.kde.org/, last access 02/2008)

the 'location' of the project where collective decision-making takes place.[21] A notable point about the mailing list is that everybody who is interested may not only follow the discussion, but can also send an email to the list and become actively involved.

The research design is based on two types of material. The communication on the mailing list is one type of material: it was analyzed for a period of twelve months. Furthermore, twelve interviews with FOSS-programmers who participated in different community driven project were conducted, transliterated and analyzed.[22] These types of material are complementary:[23] The communication on the mailing list offers an access to the public communication of the project. Here, the discussion on decision-making with its rituals and routines can be observed. The interviews were conducted with an interview guideline. They give insights in the interpretations, beliefs and normative orientations of the developers that form a common background which is not made explicit in the discussions of the developers on the mailing list.

The communication on the mailing list and the interviews were analyzed by applying qualitative-hermeneutic interpretations.[24] Three aspects of the interpretation of the material should be highlighted here. First, the chronological appearance of the communication was taken into account, following the aim to find different interpretations, and to exclude one after another in the progress of interpretation whenever inconsistency occurs. This implicates that the material was interpreted in the context of its appearance. Second, much attention was concentrated on the beginning of episodes on the mailing list,[25] since the starting sequence sets the scene for the further course of the discussion.[26] Third, it was proven systematically whether there was any empirical evidence that conflicted with the interpretation of the material.[27]

## 5 Decision-making and its implementation in a community driven FOSS project

The framework conditions of community driven FOSS projects raise the question how the participants manage to achieve an aim successfully. Following the perspective of Brunsson, it becomes clear that this question is twofold: On the one hand, one has to ask how decisions are made, and on the other hand, it has to be analyzed how (good) conditions for collective action are created. I will answer these two questions by taking a closer look at a

[21] This list can be found at: (http://lists.kde.org/?l=kmail-devel, last access 3/2008). In the course of the integration of KMail and other programs like KOrganizer, KAddressbook, and KAlarm into a 'personal information management package' the projects now share a developer mailing list. For further information see: (https://mail.kde.org/mailman/listinfo/kde-pim, last access 03/2008). The integration took place after this study had been accomplished.

[22] The communication on the KMail mailing list was observed and analyzed between 01/2001 and 12/2001. For a more detailed description of the methodology see Taubert (2006: 120-123). Two of the interviewees came from the KMail project, the other 10 developers were involved in different community driven FOSS projects. This design was chosen to compare the conclusions drawn from the KMail project with other projects for validation.

[23] See the methodical remarks in Hofmann 1999: 198.

[24] The interpretations were presented and discussed in great detail in a seminar on qualitative methods with around 10 other researchers from different disciplines including sociologists. My thanks go to the participants of this seminar.

[25] The beginning of a new episode is often (but not always) marked by a new subject-line in the emails on the mailing list.

[26] The first two aspects were borrowed from sequence analysis (Oevermann 1990: 10).

[27] This step of the analysis was inspired by grounded theory methods (Strauss/Corbin 1990: 108-109).

decision-making process on the mailing list of the KMail project. A case that provides a good starting point for the empirical analysis is a suggestion concerning the graphical user interface (GUI), that is, the graphical appearance of the program on the screen that a user would call 'the program'.

### 5.1 Argumentation and Bargaining

This discussion starts[28] with the following email by a developer who had not contributed to the project thitherto, but gained a high reputation for his work in the KDE project, of which KMail is part:

> „Hi all, ok, I have some small but important things that we (KD 7, KD 8 and I) discussed out that we need for kmail and which I like to do and need the others approval. Sorry that KD 8 changed things this week without asking and even I didn't see what was g Balancing Requirements of Decision and Action oing on." (KD 6, 2001-05-30 10:31:09)

The way in which the developer KD 6 addresses a new aim on the KMail mailing list is a bit untypical. The standard procedure is to post an email on the mailing list, and then to discuss it there, instead of announcing it as something that has already been discussed with other developers somewhere else. With the suggestion of a new aim, a decision-making situation arises and it is shaped as a selection between two alternatives: the project has to decide whether or not it is going to implement the aim.

The reference to the other developers who have already agreed on the suggestion and who began with the implementation of the feature points to the action dimension of the decision-making process. The activity of the developers indicates to the other participants that there are developers with a commitment to the suggested aim.

The fact that the developer posts an email on the list in which he asks for the approval of the others, his apologies for the action that has already been undertaken by one developer and the explanation he gives (instead of implementing the desired functionality directly) points to a first rule that has to be followed in decision-making processes in this project: aims have to be discussed first so that other developers have the opportunity to influence the decision-making process and the development-path that is followed by the project.

> „Now, what do we need and why do we need it?
>
> What we need:
>
> the default setting should be a long folder list
>
> why: because it's the common look of mail clients and other applications having a slit view." (KD 6; 2001-05-30 10:31:09)

The suggested aim is introduced rhetorically with a two-part question: one referring to the subject of the suggested development aim ('what'), and the other referring to the reason for this ('why'). The rhetorical structure of the email reflects an important aspect of the decision-making process. The normal mode for reaching a decision in this project is argumentation, that is, to convince other developers by virtue of one's arguments. From the theoretical perspective developed above, the obligation to give reasons and to discuss the aim is a norm that improves decision rationality. It allows other members of the project to participate in the discussion and to bring in other arguments, so that the argumentative basis of the decision is broadened.

The first suggestion for the new look of the GUI concerns the standard setting of the folder view on the left side of the screen: The setting can be changed by users if other settings will meet their needs in a better way. The change of the appearance is justified as an adjustment of the program to fit the look

[28] The beginning of the discussion is marked by a new subject line.

of other email clients. In the following, we shall jump to the third suggestion in order to avoid redundancy in our analysis:

> „c) more columns in the folder view and the mail view for various purposes
>
> which ones: a column in the folder view for the unread mails and one for the total mails, just like knode.[29]
>
> why: this is pretty standard and has proved to be efficient towards the user looking at the folder view, also makes the clients look more consistent.
>
> Another one in the listbox of the mails to sort threaded/unthreaded. I know that can be done via the menu or the configuration but even I had to look very hard for that feature to find it.“ (KD 6; 2001-05-30 10:31:09)

The suggestion shows the same structure as the one analyzed above. But, this time, the argument is explained in greater detail. The reference to another program exemplifies that, again, the idea for the suggestion derives from it. But mentioning the other program means more than just indicating the source of the idea for the suggestion of KD 6. Imitating the look-and-feel of other widespread and approved programs is regarded as a way to guarantee an efficient use of KMail: KD 6 connects the suggestion with an evaluation criterion, which legitimates the developmental aim, and an explanation of how the suggested aim improves the program with respect to the evaluation criterion. In other words, the aim is contextualized within a complex interpretation.[30] In the following this criterion will be named as ‘efficiency of use’ in short.

The developer closes his email as follows:

> „We would like to have these little changes done for 2.2 and would like to do them with you guys together as we think these are needed GUI improvements that would make kmail look *a lot* better and make it much easier for beginners to handle it. Please feel free to comment and blame me if something goes wrong if you’re also up with these ideas.“ (KD 6; 2001-05-30 10:31:09)

The concluding remarks give an outlook on the time schedule for implementing these features. The Code 2.2 indicates the next major release of the KDE project.[31] Referring to the date and to the developers who are ready to implement the changes, the author of the email moves from the requirements of the decision-making process (the argumentation for and justification of an aim) to the requirements of action. By describing a concrete point in time where he and his co-workers are planning to have these new aims implemented, he reduces uncertainty as he evokes the expectation that action towards the aim will be undertaken.

For the purpose of my analysis it is interesting to notice that developer KD 6 explicitly invites other project members to discuss his aims. This invitation shows that he strives toward an agreement with other developers. Furthermore, he tries to avoid unnecessary work when announcing the plan, by asking whether anybody else is already working on the implementation of these (or similar) changes.

---

[29] Newsreader for the KDE desktop, See also the website of the project: (http://kontact.kde.org/knode/, last access 04/2008).

[30] See also Holtgrewe and Brand (2007). This study applies Boltanski’s and Thévenot’s concept of ‘polity order’ to explain how new aims in FOSS projects are legitimated.

[31] The date of release was August the 15th 2001. The version was introduced as an ‘easy-to-use Internet-enabled desktop for Linux and other UNIXes’. See: (http://www.kde.org/announcements/announce-2.2.php, last access 04/2008).

The email interpreted above triggered different responses on the KMail mailings list. Moreover, it marks the starting point for a detailed discussion of the aims. Its intensity can be explained by two reasons: First, the graphical appearance of a program is an attribute of high importance as this part of the program is literally in front of every user's face. Therefore, it can be assumed that most, if not all, developers involved have a preference concerning the GUI. Second, the developer KD 6 has signaled strong commitment to the aim, and the other participants in the project have to expect that the group of the three developers will strive towards action as soon as the discussion is closed and a decision is made.

Some responses in the following discussion are questions concerning the aim leading to further explanations. However, some of the subsequent emails show disagreement. Especially suggestion 'c', the implementation of more columns, leads to controversy. One developer comments on it as follows:

> „I think you should be able to turn that off, though. I don't think it's possible with the kmail version from kde-2.1.1 to delete columns but I think that would really be a good idea. You could then add as much columns as you want without doing something wrong. You'd have to talk about the default setting though." (KD 9; 2001-05-30 10:49:42)

The developer KD 9 picks up the idea about the graphical appearance of the program but makes an alternative suggestion. Thus, from a decision point of view the decision-making process is becoming more complex and the decision rationality is improved: KD 9 does not only bring a third alternative into play (aside from leaving the GUI as it is and the original suggestion of developer KD 6), but also introduces another evaluation criterion. While KD 6 argues for 'efficiency of use' KD 9 highlights the relevance of 'adaptability' of the program for different user's needs.

From an action-rationality perspective the posting from KD 9 tends to obstruct the basis for action, as it increases the level of uncertainty. He signals commitment to his own suggestion so that it is becoming less likely that the original suggestion from KD 6 will be implemented. Besides this, the introduction of a different evaluation criterion also affects the motivational basis for action: On the one hand, the original suggestion of KD 6 cannot be regarded as a good thing, if one applies the evaluation criterion 'adaptability'. On the other hand, the suggestion of KD 9 is not desirable if one has the 'efficiency of use'-criterion in mind. Now, since the likeliness of action is reduced, it is not very surprising to see that KD 6 is unhappy with the emergence of an alternative. He argues for his initial suggestion:

> „Hmm... I think changing the default by itself without making that configurable does make the most sense. Please have a look at knode for what I mean (nsmail and outlook express do the same as pretty every mail client around) [...]
>
> What I want is to have it look like this:
>
> column1: Foldername column 2: number of unread mails column 3: number of total mails in folder. That's the precise look :)" (KD 6; 2001-05-30 11:35:09).

This reply makes another reference to the other program which offers the same functionality. More empirical evidence is given by KD 6 that the modification is widespread, and therefore makes KMail easier and more efficient to use. The second paragraph has a more illustrative character. A concrete picture is drawn as to how the GUI will look like, after the implementation is made. Again, it takes only a few moments until developer KD 9 replies to this email.

> „I don't understand why you feel that you should take the choice of what the user wants out of his hands. That is IMO[32] pretty stupid. Sure, the default is very important as most beginners don't change it but if the user KNOWS what he wants then he should be able to do it.
>
> > column1: Foldername columne2: number of unread mails column 3: number of total mails in folder. That's the precise look :)
>
> Now that I understand it I think it's a good idea" (KD 9; 2001-05-30)

Like KD 6, KD 9 argues for his evaluation criterion. He emphasizes the high relevance of 'adaptability' of the program to the needs and habits of different users. At this point, it becomes clear that the antagonism is not only about different aims, but also about different evaluation criteria that KD 6 and KD 9 apply.

Whereas KD 9 rejects the initial suggestion of KD 6 in the first part of the email, it is interesting to see that the evidence and the illustrations given by KD 6 convince him to agree on one of the changes. A third suggestion arises here that can be regarded as a compromise between the two initial ones: changing the default setting of the graphical appearance (that meets the evaluation criterion 'efficiency of use'), but at the same time making columns configurable (this meets the evaluation criterion 'adaptability' of different user's needs).

After the other developers have shown that they agree with this compromise, KD 6 pipes up again and stresses the previous decision-making process:

> „Ok, that[33] was probably too drastic. We can make it configurable with a checkbox like „use old Kmail user interface" or something
>
> >> column1: Foldername columne2: number of unread mails column 3: number of total >> mails in folder. That's the precise look :)
>
> > Now that I understand it I think it's a good idea"
>
> ok, then we agree on this as well" (KD 6; 2001-05-30 14:20:55)

Triggered by the disagreement of KD 6, KD 9 completes his suggestion with a configuration option that allows users to adapt the program to their needs. It seems that a mutual understanding has been reached, a new aim has been found, and that the decision-making process has been closed.

This first step in our analysis of a decision-making process in a community driven project, points to the following features: In decision-making processes, developers are oriented toward the norms of transparency and openness. Suggestions are open for discussion, situations in which decisions have to be made are marked as such, so that the other members of the project can participate. This orientation could already be seen in the first mail that opened the discussion. The developer KD 6 had to make excuses for having immediately begun with the implementation instead of having discussed the aim on the mailing list before. But argumentation is not just a ritual: As the decision process concerning the default setting of the graphical appearance shows, developers can be convinced by virtue of an argument.

But there is a second mechanism of closing a decision-making process. The analysis shows that aims do not only have to be suggested, but also have to be justified by interplays of suggestions, evidence, evaluation criteria, and arguments. The different evaluation criteria the proponents refer to are not taken into question but function as an anchor of the justification. In cases comparable to the one analyzed above, dissent arises with reference to these criteria. Here, it is likely that the antagonism cannot be solved by rational argumentation. Finding a compromise

---

[32] Acronym for 'in my opinion'.

[33] This refers to the initial suggestion by KD 6 to modify the GUI without implementing a configuration option.

and balancing the suggested aims and evaluation criteria on a broader bargaining level is the way to come to a decision in those cases. The discussion strives towards an absence of protest (usually uttered as 'exit' or 'voice')[34] then, in such a way that everyone involved accepts that his or her preferences are cut back.

In the light of Brunsson's distinction between the decision- and the action-dimension, the first step of the analysis yields the following results: The framework conditions of the project Kmail and the normative obligation for argumentation enhance decision rationality. The openness of the project and the opportunity to participate in the decision-making process foster the emergence of alternative aims and different evaluation criteria. More alternatives are compared, discussed, modified, and evaluated under different viewpoints.

From the collective action point of view, the results of the first step of the analysis look somewhat different. Two mechanisms that effectively reduce the numbers of alternatives could be identified: The first one is convincing the members of the project of the advantages of one alternative by argumentation. The second one is the search for a compromise which can be reached through bargaining. If a stable consensus is reached, the first solution of the decision problem connects the rationality of decision-making with the conditions for action well. It serves the requirements for collective action as it makes clear which alternative is desirable and for what reason. Aside from the motivational aspect, it also reduces the number of alternatives to a single one. Therefore, it permits clear expectations about the collective action that will be performed in the cognitive dimension.

But, as the analysis shows, not all dissent can be transferred into consensus by rational argumentation. The second solution – finding a compromise – serves the conditions for action less well. A compromise has an irrational aspect from an action point of view: Why should a developer agree with the compromise if he is not convinced that the compromise meets his evaluation criterion? Can a developer trust on the other developers' commitment concerning the compromise, when he knows that the other developers are not necessarily convinced of its superiority? It can be stressed that the compromise as an outcome is a rather weak basis for collective action.

### 5.2 The influence of reputation

Therefore, it is likely that other mechanisms help FOSS projects to manage decision-making and implementation successfully. An element of the social structure of FOSS projects is reputation and one may wonder whether reputation bears capacity for closing decisions and coordinating action. In the literature many scholars highlighted the importance of the reputation system: Developers in a project receive recognition from peers, particularly if their contributions are of high quality and have been made over a longer period of time (Lerner/Tirole 2000; Edwards 2001; Osterloh/Rota/Kuster 2002; Taubert 2006). In the long run, highly involved participants usually attain a considerable reputation. Consequently, mature projects reveal significant differences in the amount of recognition enjoyed by their participants. The observation frequently made in other, loosely coupled or loosely integrated social structures such as scientific communities, is that reputation has some coordinating capability.[35] Subsequent to this observation, the question will be addressed here, whether reputation influences decision-making processes in the case

[34] For this distinction, see Hirschman (1970).

[35] In sociology of science it is often highlighted that reputation is a basic principle for social order, as it directs attention. See for example Hagstrom 1965; Luhmann 1990; Franck 2002.

of the Kmail project. If so, how does this work? To be more precise: Do developers with a reputation for being active and productive participants in the project have more influence on decision-making than those who have less or even no reputation? Again, some hints can be found in the case of the GUI.

A highly committed member who is also the maintainer of the project[36] pipes up some hours after the compromise has been reached. After some comments on a different theme, he becomes engaged in the decision-making process about the GUI:

> "Hi, I didn't say anything about several columns. I prefer the way it is currently. When I don't have any unread mails, then I also don't need an empty column for their number. [...] At least I like to have as few columns as possible to not waste space with unimportant things." (KD 10; 2001-05-30, 18:09:49)

By stating that he has not said anything about the suggested aim, KD 10 positions himself as a relevant player in the decision-making process. This positioning is marked by the 'hi' which is not located at the beginning of the email but rather in the middle. He does not regard the decision-making process as being closed in this passage (and the reaction of KD 6 shows acceptance of this positioning), and it becomes apparent that KD 10's agreement is considered to be highly relevant for any decision-making in the project.

His contribution to the discussion shows that even developers with a high reputation, and the position of a maintainer of the project, cannot reject a suggestion right away only by virtue of his reputation or his position. The fact that he formulates a proper argumentation suggests the interpretation, that neither reputation nor high involvement in the project frees developers from the obligation to give proper reasons for their points of view. Compared with reputation, the obligation to give reasons for a viewpoint is the more fundamental principle.

In his response to this email KD 6 refers explicitly to KD 10's role as maintainer of the project:

> „Yes, well, agreed you're the maintainer, that gives your personal preference a great influence in the behavior [of the program, N.C.T]. I agree with you that this might be true for some users, especially long-term kmail and unix users. But if you want to get windows users to use it, the default has to be different and, most important, consistent with knode which orients itself on the „standard" user interface." (KD 6 2001-05-30 18:53:26)

Although the developer affirms that a maintainer is a relevant player in a decision-making process, KD 6 does not behave in a way that is different than in situations of dissent with other developers (e.g. the situation analyzed above). He takes note of the disagreement, but does not give up his suggestion as one might expect. Instead, he begins to give reasons for it again. In this email he frames his argument in a slightly different way. The imitation of the appearance of other programs makes it easier for beginners to work with KMail. Aside from the evaluation criterion 'efficiency', there is another one that can be named 'market share' or 'market success' of the program.

What can we learn about the influence of reputation on decision-making processes from this example? The intervention of the project's maintainer takes place at a point in time at which the protagonist KD 6 is trying to move from decision-making to the implementation and it has the same effect the dissent between KD 6 and KD 9 had above. It increases the degree of

[36] For my argumentation it is not important that the developer is also the maintainer of the project. In the other cases on the KMail mailing list other developers with high reputation caused a reopening of the argumentation that already seemed to be closed.

uncertainty and obstructs the conditions for action on the cognitive, motivational and commitment-related dimension. Neither KD 10's high reputation nor his role as a maintainer lead to the rejection of the suggestion but to a rehashing of the argument, with its positive effects for decision rationality and negative effects for the conditions for action. In other words, the example suggests that in the project KMail neither reputation nor hierarchy play a decisive role in paving the way toward collective action.

But this is not the only conclusion that can be drawn from the analysis. The reopening of the decision-making process after a situation in which a compromise seemed to have been reached, suggests that the maintainer of a project is regarded as a relevant actor in respect to decision-making by other participants. He is treated as an actor with whom an agreement has to be reached.

This finding supports a common observation in the literature, namely that reputation is a precondition for influencing the decision-making process in the sense that the respective actor is included in the discussion process and that his arguments are taken into account (Brand/Holtgrewe 2004: 14; Taubert 2006. 172 ff). The more generalized hypothesis that should be investigated on empirical grounds would be: Reputation influences decision-making as the consideration of an argument depends on the extent of reputation the respective actor enjoys.

### 5.3 Indecisiveness of community driven FOSS projects?

So far, the initial question of how community driven projects manage to make decisions and implement them remains unanswered in cases where an agreement cannot be reached by argument and a compromise cannot be found. In those cases the development process could easily stagnate. The evidence given above suggests that, in the case of Kmail, neither hierarchy nor reputation will help in those situations. Therefore, one could expect that controversies continue for a long time without a possibility to solve them. Therefore, one could assume that community driven FOSS projects like Kmail struggle with a certain weakness or even indecisiveness of decision-making. But, in fact, this kind of situation rarely emerges in the analyzed case, since two non-communicative elements operate silently in the background. They prevent stagnation and help break down blockades.

Therefore, I conclude the analysis of the GUI and provide some evidence for these elements from two interviews conducted in this study. One KMail developer describes the factors that prevent a project from running into blockades. When asked if dissent about aims leads to trouble and block the development, the developer answered:

> „No, not in the long run. Well there would be a thread of 50 emails or so. [...] That might go on for one and a half weeks in an extreme case. It goes on and on until people are in such a snit that they get it all together and implement something. It may well be that the one or the other isn't happy with it afterwards, but you can't please all the people all the time." (KD 1, interview)[37]

This quotation confirms the analysis above, that there are cases of dissent, which cannot be solved by argumentation. The developer describes that participants come to a point at which they get tired of discussing the issue, break up the argumentation, and start to implement something. One can say that, in situations of enduring dissent, time helps to come to a solution as participants are aware that stubbornly insisting on one's own point of view – repeating arguments, providing more evidence and reformulating evaluation criteria – neither helps the decision-making process nor its implementation

[37] The interviews were originally conducted in German, the quotations in this section are translated by the author.

in collective action. I would like to suggest that such an increase of a pragmatic willingness to act should be interpreted with reference to a framework condition of the project. It seems reasonable to assume that the willingness to come to a solution is very strong in projects, where an intrinsic interest serves as an eminent motivation. In these cases stagnation deters the developers from developing software, viz. an activity they are very much interested in.

Aside from consensus and compromise, there is a third way to come to a decision that can be found in the interviews. The following passage from another KMail developer illustrates this:

> "Most importantly, there is no one (in FOSS projects, N.C.T.) who really says how the work has to be done if the project can't decide. In the extreme case, it is the one who opens the editor and writes down the patch. The one who does the work and not the one who babbles on and on." (KD 2, interview)

This quotation gives evidence that a stagnation of the development process can occur, and that there is no decision maker who can decide top-down in a hierarchical manner. Instead, the lack of a legitimized decision maker who decides in the case of dissent is compensated by another mechanism, which is the opportunity to switch over from decision-making to action without having reached an agreement in the project. It is interesting to note that the developer describes the development activity very demonstratively and colorfully with terms such as 'doing the work' and 'opening the editor', whereas participation in the discussion is referred to in quite disrespectful terms. Contrasting these two kinds of involvement shows that practical development work is held in higher esteem than participation in the argument. But as seen in the case analyzed above, the argument is a crucial factor: It is necessary to discuss the suggestion before switching to the development activity. Remember that the developer KD 6 had to apologize because the development activity already started without any prior discussion.[38]

## 6 Conclusion

Collective decision-making and implementation in FOSS projects take place in a constellation of conflicting demands. On the one hand, a larger number of developers being involved improve the search for solutions (Kuk 2006: 1034). On the other hand, a larger number of developers complicate the process of reaching a decision. This does not only lead to an increase of communication and cooperation costs (Brooks 1975) but, with reference to Brunsson, it also increases uncertainty, and can obstruct the basis for action. In this analysis three outcomes of the decision-making and implementation problem could be identified in the case of Kmail. Therefore, in this closing section the different outcomes will be discussed in the context of the theory developed above. In addition, it will be evaluated how they match the requirements of decision-making and action.

(a) *Rational consensus* as an outcome seems to match the requirements of decision-making and the requirements of action well. Closing the

[38] An often discussed result of dissent is forking a project and developing different versions in separate projects. In this case study such a dramatic change of the project structure could not be observed and it seems that forks seldom happen. There are two factors that stand against forks. First, in community driven projects splitting a project would also imply to split-up manpower. This would increase the workload for each participant, slow down the speed of the development progress and could lead to the necessity to cut down the project's aims. Second, it is likely that incompatibilities between the different versions of the program would arise. This effect is critical in cases where software with large network effects is developed. The negative impact of those events is well known to FOSS-developers from the history of the UNIX operating system (see McKusick 1999).

decision-making process by virtue of an argument after having different suggestions discussed in-depth, leads to a well-founded decision. After a decision is made, it is clear what kind of action has to be expected on a cognitive level and for what reason the action is desirable. Those circumstances should lead to a high level of motivation among the developers. And it is also likely that the protagonists of the chosen alternative have expressed commitment to the aim during the discussion (like in the example above), so that a good basis for collective action should be created.

The only critical aspect of this solution of the decision problem is a considerable degree of uncertainty that can emerge during long-lasting discussions. I would like to suggest that this relatively high level of uncertainty, allowed in the course of a decision-making, should be understood with reference to the motives of the developers to participate in FOSS projects. As stated above, they are intrinsically interested in the development process itself and in the success of the project, and they can expect that other developers share this attitude. These framework conditions might permit a higher level of uncertainty than in other organizations where such conditions do not exist.

(b) *Compromise* matches requirements of rational decision and action less well than rational consensus. When a compromise is introduced in the decision-making process, the developers have usually discussed the suggestions in detail. Therefore, it is unlikely that new arguments will be pushed forward and the rationality of the decision will be improved by further discussion. From the requirements of action the compromise reveals a particular irrationality: Why should a developer participate in the implementation of a certain compromise, although it is only second choice for her or him, and not the right thing to do? If the lower degree of motivation of part of the developers is taken into account, it is supposable that he or she accept that other developers work on the implementation of the aim, but is not getting involved in the work him- or herself. In other words, a compromise is a solution for the problem of decision-making as it marks an end of a discussion that tends to become unfruitful. But it nevertheless is inclined to obstruct the motivation for *collective* action if some developers think that there are better ways to go.

(c) *Moving from decision toward individual action* is the last solution for the problem of decision-making and its implementation in FOSS projects. The idea of collaborative work is abandoned here as it is foreseeable that only the (group of) developer(s) who regard(s) the option as the right thing will contribute to the process of the implementation. Since a basis for collective action cannot be created, individual action takes its place. From the viewpoint of decision rationality this option is also not preferable: It might happen that the developer who moves from decision towards individual action only takes his own suggestion, arguments, and evaluation criteria into account, so that the final level of rationality of the decision is low. Thus, individual action seems to be the worst way to deal with the problem of decision-making and action in FOSS projects. But from the viewpoint of a social structure aiming to develop software, there is one situation that should be avoided at all costs: To be stuck in the development process for a longer period of time.

## 7 References

Bonaccorsi, Andrea/Cristina Rossi, 2003a: *Licensing schemes in the production and distribution of Open Source software. An empirical investigation.* <http://papers.ssrn.com/sol3/papers.c

fm?abstract_id=432641> (last access 04/2008).

Bonaccorsi, Andrea/Cristina Rossi, 2003b: Why Open Source Software can succeed. *Research Policy* 32 (7), 1243-1258.

Brand, Andrea/Ursula Holfgrewe, 2004: *KDE im Kontext: Open Source Software Entwicklung und öffentliche Güter*. <http://www.uni-due.de/imperia/md/content/soziologie/ abuh-indsoz-604.pdf> (last access 04/2008).

Brooks, Frederic, 1975: The Mythical Man Month: Essays on software engineering. Reading: Addision-Wesley.

Brunsson, Nils, 1985*: The Irrational Organization. Irrationality as a Basis for Organizational Action and Change*. Chichester et al.: John Witley & Sons.

Cohen, Michael D./James G. March/Johan P. Olsen, 1972: A garbage can model of organizational choice. In: Administration Science Quarterly Vol. 17 (1), 1-25.

Cyert, Richert M./James G. March, 1963: A behavioural theory of the firm. Prentice-Hall: Englewood.

Edwards, Kasper, 2001: Epistemic Communities, Situated Learning and Open Source Software Development. Technical University of Denmark: Working Paper. <http://opensource.mit.edu/papers/kasperedwards-ec.pdf> (last access 04/2008).

Elster, John, 1986: Rational Choice. Oxford: Blackwell.

Franck, Georg, 2002: The Economy of Attention. *Scientometrics* Vol 55 (1), 3-26.

Gläser, Jochen, 2006: *Wissenschaftliche Produktionsgemeinschaften. Die soziale Ordnung der Forschung*. Frankfurt/New York: Campus.

Gosh, Rishab Aiyer/Gregorio Robles/Ruediger Glott, 2002: Free/Libre and Open Source Software: Survey and Study. D 18: Final Report. Part V. Software Source Code Survey. <http://www.infonomics.nl/FLOSS/report/> (last access 04/2008).

Grassmuck, Volker, 2002: *Freie Software zwischen Privat- und Gemeineigentum*. Bonn: bpb.

Hagstrom, Warren O., 1965: *The Scientific Community*. New York/London: Basic Books.

Hardin, Garret, 1968: The Tragedy of the Commons. In: *Science* 162 no. 3859, December 1968, 1243-1248.

Hertel, Guido/Sven Niedner/Stefanie Herrmann, 2003: *Motivations of software developers in Open Source projects: An Internet-based survey of contributors to the Linux Kernel.* In: *Research Policy* 32 (7), 1159-1177.

Hirschman, Albert O., 1970: *Exit, voice and loyality: Responses to decline in firms, organizations, and states*. Cambridge: Harvard University Press.

Hofmann, Jeanette, 1999: „Let a thousand proposals bloom“ - Mailing-Listen als Forschungsquelle. In: Bernad Batinic/Andreas Werner/Lorenz Gräf/Wolfgang Bandilla (eds.), *Online Research. Methoden, Anwendungen und Ergebnisse*. Göttingen et al.: Hogrefe.

Holtgrewe, Ursula/Andreas Brand 2007: Die Projektpolis bei der Arbeit. Open Source Software-Entwicklung und der neue Geist des Kapitalismus. In: *Österreichische Zeitschrift für Soziologie*, Vol. 32 (3), 25-45.

Holtgrewe, Ursula/Werle Raymund 2001: De-Commodifiyin Software - Open Source Software Between Business Strategy and Social Movement. Science Studies 14 (2), 43- 65.

Koch, Stefan/Georg Schneider 2002: Effort, co-operation and co-ordination in an open source software project: GNOME. *Information System Journal* 12 (1), 27-42.

Kuk, George 2006: Strategic Interaction and Knowledge Sharing in the KDE Developer Mailing List. *Management Science* 52 (7), 1031-1042.

Lakhani, Karim/Robert G. Wolf, 2005: *Why Hackers Do What They Do: Understanding Motivation and Effort in Free/Open Source Software Projects. MIT Sloan School of Management.* <http://ocw.mit.edu/NR/rdonlyres/Sloan-School-of-Management/15-352Spring-2005/D2C127A9-B712-4ACD-AA82-C57DE2844B8B/0/lakhaniwolf.pdf> (last access 04/2008).

Lerner, Josh/Jean Tirole, 2000: *The Simple Economics Of Open Source*. Working Paper 7600. Cambridge: National Bureau of Economic Research. <http://www.hbs.edu/research/facpubs/workingpapers/papers2/9900/00-059.pdf> (last access 04/2008).

Lerner, Josh/Jean Tirole, 2001: The open source movement: Key research questions. In: *European Economic Review* 45 (4-6), 819-826.

Lerner, Josh/Jean Tirole, 2002a*: The Scope of Open Source Licensing*. Harvard Buisness School Working Paper. http://www.people.hbs.edu/jlerner/OSLicense.pdf (last access 04/2008).

Lerner, Josh/Jean Tirole, 2002b. Some Simple Economics of Open Source. In: *Journal of Industrial Economics* 50(2), 197-234

Luhmann, Niklas 1990: *Die Wissenschaft der Gesellschaft*. Frankfurt a.M.: Suhrkamp.

March, James G./Herbert A. Simon, 1958: Organizations. New York et al.: Wiley.

March, James G., 1994: A Primer on Decision Making. How Decisions Happen. New York: Free Press.

March, James G., 1978: Bounded Rationality, Ambiguity, and the Engineering of

Choice. In: *The Bell Journal of Economics* Vol. 9 (2), 587-608.

O`Mahoney, Siobhán, 2003: Guarding the commons: how community managed software projects protect their work. *Research Policy* 32 (7), 1179-1198.

McKusick, Marshall Krik, 1999: Twenty Years of Berkeley Unix: From AT&T-Owned to Freely Redistributable. In: Chris DiBona/Sam Ockman/Mark Stone (eds.), Open Sources. Voices from the Open Source Revolution. Beijing et al. O'Reilly.

Oevermann, Ulrich, 1990: *Klinische Soziologie. Konzeptualisierung, Begründung, Berufspraxis und Ausbildung.* <http://publikationen.ub.uni-frankfurt.de/volltexte/2005/534/pdf/KlinischeSoz-1990.pdf> (last access 04/2008).

Osterloh, Margit/Sandra Rota/Bernd Kuster, 2002: *Open Source Software Production, Climbing on the Shoulders of Giants.* <http://opensource.mit.edu/papers/osterlohrotakuster.pdf> (last access 04/2008).

Raymond, Eric S., 1999: *The Cathedral & the Bazaar. Musings on Linux and Open Source by an Accidental Revolutionary*. Bejing et al.: O'Reilly.

Shah, Sonali K., 2006: Motivation, Governance, and the Viability of Hybrid Forms in Open Source Software Development. In: *Management Science* 52 (7), 1000-1014.

Simon, Herbert A., 1957: Models of Man. Mathematical Essays on Rational Human Behaviour in a Social Setting. New York: John Wiley.

Stallman, Richard M., 2002: *Free Software, Free Society. Selected Essays of Richard M. Stallman.* Boston: GNU Press.

Stallman, Richard M., 1999: The GNU Operating System and the Free Software Movement. In: Chis DiBona/Sam Ockman/Mark Stone (eds.), *Open Source. Voices from the Open Source Revolution*. Beijing et. al.: O`Reilly, 53-70.

Strauss, Anselm/Juliet Corbin, 1990: Basics of Qualitative Research. Grounded Theory Procedures and Techniques. London et al.: Sage.

Taubert, Niels C., 2006: *Produktive Anarchie? Netzwerke freier Softwareentwicklung*. Bielefeld: transcript.

von Hippel, Eric, 2001: *Open Source Shows the Way: Innovation by and for Users – No Manufacturer Required.* <http://ebusiness.mit.edu/research/papers/133%20von %20hippel%20OSS%20innovation.pdf> (last access 04/2008).

von Krogh, Georg / Eric von Hippel 2006: The Promise of Research on Open Source Software. In: *Management Science*, Vol. 52 (7), 975-983.

Weber, Max, 1972: Wirtschaft und Gesellschaft. Grundriß der verstehenden Soziologie. 5. rev. Aufl. Tübingen: Mohr.