

Framework for Governance in Open Source Communities

Christoph Lattemann, Stefan Stieglitz
 Potsdam University
 August-Bebel-Str. 89, 14482 Potsdam
lattema@rz.uni-potsdam.de / stieglit@rz.uni-potsdam.de

Abstract

In recent years, the development of software in open source communities has attracted immense attention from research and practice. The idea of commercial quality, free software, and open source code accelerated the development of well-designed open source software such as Linux, Apache tools, or Perl.

Intrinsic motivation, group identification processes, learning, and career concerns are the key drivers for a successful cooperation among the participants. These factors and most mechanisms of control, coordination, and monitoring forms of open source communities can hardly be explained by traditional organizational theories. In particular, the micro and macro structures of open source communities and their mode of operation are hardly compatible with the central assumption of the New Institutional Theory, like opportunistic behavior.

The aim of this contribution is to identify factors that sustain the motivation of the community members over the entire life cycle of an open source project. Adequate coordination and controlling mechanisms for the governance in open source communities may be extracted.

1 Introduction

The software industry is dominated by a strong competitive influence. Recent years have been characterized by a displacement of software companies, resulting in segmentary oligopolistic or even monopolistic structures. It is therefore all the more impressive that a sustainable and successful development of open source communities (OSCs) has been possible, even as open source communities are characterized as non-profit organizations [35].

The reason for participating in these kinds of communities as an active worker can not be explained by monetary incentives. Intrinsic motivation [59] and learning [65] are the key drivers in this context. In addition, non-monetary extrinsic motivations, like reputation, group identification processes [52, 20], and career plans [32] are most important for the successful operation of open source communities.

A successful operation of net-based organizational units, like in OSCs, has to consider that rules, motivation, and incentive structure not only allow a coordination of contributions, but also increase the involvement and thus

inspire people to create contributions. Traditional agency-based governance approaches [10], which focus on monitoring, control, and supervision of internal processes, are often neglect this point¹. In most cases these approaches stress monitoring systems, incentive contracting, or the use of sanctional mechanisms with a mainly monetary aspect. In particular, the micro and macro structures of open source communities and their mode of operation are hardly compatible with these kinds of governance instruments and with the central assumption of the prevailing New Institutional Theory, such as opportunistic behavior. These approaches and their implications for options for actions will not work.

Although considerable research has been devoted to the growth and expansion of open source communities and the comparison between the efficiency of corporate structures and community structures in the field of software development [1], rather less attention has been paid to their governance structures (control, monitoring, supervision). Especially psychological and sociological aspects must be considered. The Stewardship Approach, developed in the 1990s, shows some starting points for this discussion [13]. Donaldson and Davis are assuming a different image of human behavior. Workers are motivated by intrinsic incentives and not influenced by opportunistic behavior. Following this, trust [15] and intrinsic motivation are the key drivers for a new coordination system which is used by OSCs.

In this context, psychological studies from the mid-1980s on virtual communities show communication and collaboration in computer-mediated environments as being typically rather anomic [23], less tolerant [18], and absent of transferable behavior [55]. Thus, the topic of governance in open source communities faces a new (mediated) organizational culture from the viewpoint of business management sciences.

Additionally, new analyses show that open source communities follow a life cycle [58, 67]. The internal organizational structure is changing by going through different life cycle stages. The implementation of hierarchic structures, which can be observed in practice, ensures that there are adequate controlling and coordination mechanisms among all organizational units [1].

¹ Control, monitoring, and supervision mechanisms are collectively defined as governance in the remainder of this paper.

The aim of this article is to examine central structures and coordination patterns in open source communities. These results allow further studies and help open source members to understand the role of governance within their community. Therefore, the evaluation is based on a systematic review of relevant literature and empirical studies related to open source communities and to virtual organizations. The focus is to identify mechanisms that work as open source-specific governance tools. One of the most important characteristics of these tools is that they are of a non-monetary nature. Therefore the question has to be answered by which incentives the absence of money is compensated. Subsequently, a heuristic approach is presented to categorize open source projects in order to derive the need for governance actions.

All control, monitoring, and supervision structures are gathered in the underlying definition of governance of this paper. Therefore, chapter 2 examines the specific conditions for the governance in OSCs.

Chapter 3 analyzes which motivation factors are important to particular member groups of the OSCs in different life cycle stages, to create a positive added value. By knowing more about motivation factors, mechanisms and procedures can be developed and implemented to increase organizational efficiency. To examine this, an individual, situation-based, and behavioral approach, which considers micro-organizational aspects as well, is developed, while traditional governance theories apply a holistic view of stakeholders.

This contribution is based on the assumption that open source actors differ in their motivation and that these motivations are related to their function and duties within the community.

Therefore, we show a classification of different open source actors on the basis of functional characteristics. In addition, we consider different motivation factors, referring to life cycle stages of open source projects.

Additional information from analyses of project life cycles and the roles of governance tools in specific life cycle stages will be discussed. Finally, a summary is provided and open research questions are named.

2 Conditions for Governance in Open Source Communities

2.1 General Characteristics of Open Source Communities

Open source software differs from commercially produced software among other criteria in the following ways [1, 58]:

- the permission for free propagation of software,
- the free availability of the source code,
- the right to change the source code, and

- the free propagation of the software license for anyone who wants to use the program.

In most cases these rights are guaranteed by the use of the GPL (General Public License) [18, 58]. As a legally binding contract, the GPL guarantees that all software containing parts covered by the GPL are themselves also subject to the GPL and must fulfill the criteria mentioned above [47].

This basic concept of the development of free software and open source code on the basis of volunteers not financially remunerated directly influences the organizational structures.

OSCs differ from common enterprises in their coordination and organizational structure. The work is done on a voluntary basis, and there are no guidelines regarding time and intensity of work.

Due to the decentralized and computer-based value creation process, personal contacts while working on the project take place only on a small scale. Despite these differences OSCs succeed in manufacturing marketable and competitive products (e.g. Linux, Apache, Mozilla, etc.).

This is made possible by characteristics which are featured by software development process in general. Software development differs from the production of other goods, such as manufactured products and most services, in the following ways:

- A common product is developed by a community or organization, but the individual parts are produced (geographically) decentralized. Communication and coordination of the project are based on modern communication technologies which makes teamwork possible for members worldwide.
- The contributions required for further development of the software are sequential, meaning innovation is based on previous development and does not making radical leaps. This kind of work can then be characterized as complementary because a rising number of solutions develop synergies for the optimization of the product [47].
- Software programming can be divided into subprojects, which can continue to be developed independently (granularity).
- Software can be development with no chronological order, meaning that elements or modules which are programmed later can be integrated into already active software (modularity) [58].

These criteria of the software development process in general and open source products in particular fulfill the necessary conditions which offer the participants of the project the most decision-making freedom. They can decide on their own in which part of the project they will become involved, where they complete their work, and

what resources they will contribute. The high degree of self-determination of each individual user that is based on granularity and modularity is essential for the OSCs to maintain the motivation of volunteer workers. Thus, self-determination can be understood as a fundamental prerequisite for governance in open source environments.

There are basically two sources in OSCs which motivate open source members to participate [11, 17, 47]: The first is motivation of an extrinsic nature, which is quite important in profit-oriented companies and is adopted there as a dominant control tool. Extrinsic motivation means engaging in an activity to receive rewards for the activity. These rewards can be of a material nature, like money, or of a social nature, like growing reputation or prestige. Especially Lerner and Tirole attribute much individual motivation to reputation building [32]. A second source for contributing may be based on intrinsic nature, i.e. engaging in an activity out of pure pleasure.

Beyond these core motivators in open source projects, further incentive mechanisms can be identified, which focuses on the intrinsic as well as extrinsic motivation of coworkers.

While it can be assumed in the context of monetary incentives that each individual is generally interested to a certain degree in maximizing his financial resources, this does not apply for different, non-material incentives. It is therefore necessary to take a close look at the individuals summarized in an open source community and to categorize them in order to determine their motives more thoroughly on the micro level.

Through the identification of clear functions in open source communities it is possible to recognize situation-specific motives and to activate them purposefully by governance instruments.

2.2 Participants and Working Groups of Open Source Communities

OSCs integrate heterogeneous member groups, who have different tasks within open source projects [1, 22, 19]. In reality, individuals can not be fixed to one specified member group. Usually one individual fulfills functions in different groups and is therefore a member of several of these categories.

On an aggregated level, three typical member groups can be identified: bug fixers, programmers, and coordinators.

“**Bug fixers**” contribute irregularly and rarely to the open source organizations. They are actually software users who communicate errors to the community. Raymond assumes that this group represents on average more than 75% of all open source members [52]. An essential component of bug fixing is finding software errors and deficiencies and communicating these to programmers and managers. Furthermore, many bug fixers provide solutions to the problems they find. Bug fixing is an essential aspect for success in OSCs.

Especially the beta testing phase, when bug fixing occurs, uses many organizational resources in software development. The acceptance of software is fundamentally influenced by the intensity of beta testing [47, 52]. Through constant checking and adjusting, software can be modified and optimized during the creative process [1].

Optimizing open source software for better usage is the most important incentive for bug fixing [21]. Other motivational aspects such as increasing reputation are supported by the publication of the names of the most active bug fixers [33, 41]. These motivational factors exhibit a non-monetary extrinsic nature. They can therefore not be controlled directly by management of open source organizations, in contrast to management in profit-oriented companies.

The targeted use of governance instruments can increase the commitment of bug fixers, if these instruments are adequate and specific.

Because an application of inadequate governance instruments may crowd out intrinsic motivation, while extrinsic incentives may be stimulated. Overall, the level of motivation may decrease.

Programmers, the second group of community members, develop OSC software by contributing their knowledge. In many cases programmers worked as bug fixers at the beginning of open source projects. Programmers are characterized by working more regularly and more frequently than bug fixers. Dempsey estimated that 8.5% of all open source workers belong to this category [12].

Lerner and Tirole assume that one core motivational aspect to programmers comprises of signaling knowledge to potential employers of profit-oriented companies [32]. By working on open source projects, programmers have the opportunity to convey specific knowledge to people outside the project. Therefore, programmers participate not only because of a strong intrinsic motivation but also because of extrinsic aspects.

Programmers, in contrast to bug fixers, are much more integrated in organizational coordination and communication processes. Governance tools should ensure that there are coordination mechanisms which facilitate working processes and communication among other programmers, bug fixers, or managers.

Founders and programmers in open source projects often change roles as a system grows to **managers**. Their tasks are extended to a strategic focus, such as implementing coordination mechanisms. Often these managers are elected by the community or by some of the registered community members². Elections guarantee reputation and prestige within the community. Often

² E.g. Apache board of directors is elected by members who are proposed to be “Foundation Members” of Apache organization [see <http://apache.org>].

managers are paid employees in companies or universities and are assigned to work on open source projects [53].

The motivation of managers as well as of programmers can be traced back to career plans. This is a reason why investing in reputation is an important factor for open source managers. Therefore, in some ways, financial incentives and strategic decisions gain in priority.

The classification of these three important member groups shows the heterogeneity found in open source communities of motivational aspects and reasons for participating. It is necessary to consider these characteristics when examining the usability of governance tools. It must be determined whether coordination structures, adequate to all groups, can be implemented, or whether these structures should be applied to single groups.

2.3 Life Cycle Stages in Open Source Communities

To assess the situation-based and adequate application of governance tools for controlling community members and contribution level, it is not enough to consider various member groups. Because of changes caused by different stages of project life cycles, it is possible that motivational basis of member groups could diverge or clash, which may lead to stagnation of project development.

The identification of life cycle stages is often based on revenue or sales numbers in conventional companies. However, open source organizations do not have these figures because no product has been sold. Therefore, the only possibility is to use an auxiliary variable such as the number of downloads of open source software. Variations of downloads can be used as one important indicator to identify individual life cycle stages.³ A continuously increasing number of downloads suggests a growing interest among software users [67]. Other indicators suggested by Schweik and Semenov, such as (1) annual growth of participant base, (2) annual growth of user community, (3) growth in “market share,” (4) user satisfaction with the product, and (5) peer recognition of the product, can be used as a measure of project success as well. However, these criteria are quite difficult to measure [58].

Scientific literature differs between three and eight project life cycles [67, 34, 58]. For the explanation of different governance actions in OSCs, it seems appropriate to concentrate on four clearly specifiable life cycle stages (see fig. 1):

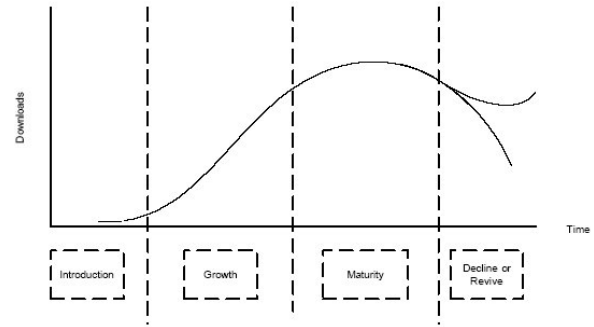


Figure 1: Life cycle stages [67]

1. Introduction
2. Growth
3. Maturity
4. Decline or Revival

The **introduction stage** is characterized by the generation of ideas by the founders. The core group negotiates an informal structure consisting of relatively general roles for each member. Trust is the key factor in the coordination process. All participants are mainly intrinsically motivated in this stage, while extrinsic incentives hardly exist. The fun of programming, conversion of creativity, and noticeable software advancement are key drivers for motivated workers [64]. Most important to enter the next stage is to attract more developers to make sure that a critical mass of community members is reached. This allows a constant development of the open source software [58].

Because of the increased size and the transient nature of the team membership, there is a need for a more formalized structure in the **growth stage**. Reliance on information and communications technology (ICT) such as concurrent versions systems (CVSs), discussion groups, and mailing lists becomes important for coordinating the efforts of the community as can be seen in complex projects like Linux, Apache, or Mozilla. By riding the life cycle the tasks of the founders change. They may assume the role of “accelerators,” employing systems and structures to enable and manage the project’s growth [66, 58]. At this stage, the increased amount of work allows members to choose their own more specialized roles such as code tester, release manager, interface designer, support manager, documentation writer, or bug fixer. The resultant structure is still relatively centralized, with the core developers retaining overall project control, but lesser functions being delegated to others outside the core group. Using a more specialized work model has the consequence of splitting member groups. Each of these previously mentioned groups contributes due to other project motivational reasons.

In the **maturity stage**, the number of users (combined with the number of downloads) and developers reaches its

³ Most open source projects and number of downloads are published at <http://sourceforge.net>.

maximum [67]. The importance of trust as a coordination and controlling instrument decreases while extrinsic factors become more important. A high degree of delegation and self-management leads to a very high level of task specialization.

The central focus of the administrative core group during this stage is to sustain the project [66]. Because of the larger project size, there is a need for highly sophisticated mechanisms to coordinate and control the diverging structure [54].

Decline stage or revival stage is marked by decreasing users and developers. There are fewer downloads of the product as users become less interested in the project. This has different possible reasons, such as better competing systems or departure of founding members. Another reason may be that managers want to influence product development in ways different from the community. To avoid the slowing or stagnation of software advancement, it is necessary to generate a reconcentration of diverging concerns [67].

It can be deduced that many open source projects are affected by these heterogeneous changes of motivational aspects in each member group. Bug fixers and programmers are not integrated in management tasks, and they are basically intrinsic motivated. As shown, they are extrinsically motivated too, but even this is a non-monetary factor. This may be different from the case of members with managerial responsibilities. They likely want to earn money over the long term which may lead to decisions not compatible with the interests of other community members.

These differences in goals and motivational incentives may lead to the collapse of a project. This can be reduced by the implementation of adequate control, supervision, and monitoring instruments.

2.4 Motivation and Objectives of Member in Changing Life Cycle Stages

The effectiveness of governance tools is influenced by other aspects, such as the number of members or the complexity of the software, which are determined by life cycle stage. There is therefore an interaction between these factors and the motivation and participation objectives of members [see table 1]. Schweik / Semenov and Wynn describe different life cycle stages but they do not explicitly discuss changing motivation of member groups within these stages [58, 67].

Motivation	Bug fixer	Programmer	Manager
Introduction			intrinsic
Growth	extrinsic	intrinsic / extrinsic	mainly intrinsic
Maturity	extrinsic	intrinsic / extrinsic	mainly extrinsic
Decline or Revival	extrinsic	intrinsic / extrinsic	Revival: intrinsic Decline: mainly extrinsic

Table 1: Motivation of member groups in different life cycle stages

In the growth stage extrinsic aspects such as reputation, career planning, or increasing knowledge become more important [33]. This changing motivation is influenced by a modified objective for working on the open source project. In the growth of maturity stages, improving software is no longer the dominating objective for contributing ideas; it is just an intermediary to acquiring more members, knowledge, and reputation [33]. Finally, these objectives lead to improved career chances for individuals.

However, Osterloh states that the identified extrinsic factors do not suffice to explain the phenomenon of open source communities [47]. Therefore, it can be assumed that all member groups are partially intrinsically and partially extrinsically motivated in every stage of the life cycle. Numerous analyses show that open source members have fun while programming. They feel a kind of self-realization while doing creative work and supporting a useful project [1, 28, 29]. However, intrinsic motivation is related to the degree of self-determination. The feeling of self-determination within OSCs is confirmed in two ways. The first is high granularity, which allows programmers to decide what kind of work they want to do. The second is a high degree of modularity, which offers many possibilities to participate in improving the software [7, 27, 58].

In the special case of open source development, different reasons for participating in a project can be combined in some ways as long as there is a common main objective to improve the open source product. However, it is possible that different objectives are not compatible, which may lead to governance problems. In principle, the main objective to improve the program may be deduced in each member group as follows:

- Bug fixers participate in open source projects to obtain improved software for their own use [21, 33].
- Programmers try to achieve a higher internal and external level of reputation and want to improve their career chances [47]. Additionally, they want to optimize software as long as they are intrinsically motivated (e.g. fun) [52].
- Managers want to increase their reputation as well, but they are also seeking monetary

rewards. Both objectives are connected to the improvement of the software and an increase in the quality of the product [48].

3 Governance in Open Source Communities

3.1 Application of Conventional Governance Mechanisms in Open Source Communities

The theoretical root of this paper is based on the approach of the *organizational control* [48], where governance is understood as a toolbox for control, supervision and monitoring. In this context, governance must achieve motivation and converge different objectives of all member groups.

Conventional theories of governance are based on a holistic view of companies and their organizational units. These theories assume that there are only a few or even a single quantifiable objective of all stakeholders. Furthermore, it is assumed that every stakeholder is interested in maximum or optimum achievement of objectives. In contrast, our behavioristic approach assumes that there are many individual and changing objectives. This was shown in the preceding discussion about life cycle stages and drivers for motivation. A main problem in implementing governance tools is considering which objectives exist for different members. Profit oriented companies solve this problem by defining one main objective, e.g. profit maximization, which, through support from monetary incentives, is accepted by all member groups. This procedure cannot work successfully in an organization which is based on volunteer work. The absence of a common main objective may lead to conflicts of interest in and among member groups. Governance mechanisms are implemented to avoid and solve these conflicts in conventional companies. Furthermore, governance instruments are used to increase organizational efficiency and fulfill monitoring tasks, e.g. communication policies or conflict management rules. Employees follow these rules to avoid financial sanctions or being laid off.

The effectiveness of sanction mechanisms in OSCs may be questioned because there is no formal or existential dependence of the members on the open source project. That is why the options of effective sanction mechanisms are limited to the exclusion of counterproductive members and a loss of reputation. Practically speaking, members that violate organizational policies may be "flamed," meaning they are publicly named and judged by other community members [24].

By implementing hierarchic structures, managers try to assure that coordination and communication within the community is optimally configured. However, this assumes that organizational policies are accepted by most of the community members. Profit-oriented companies do

not have a problem at this point because policies and hierarchies are legitimated by company owners. Within OSCs it is necessary that managers have an adequate reputation to be accepted as leaders. For example, board members of the Apache community are elected by foundation members. "Individuals who have made sustained and important contributions to one or more of the foundation's projects" can be nominated to become foundation members [4].

Thus, although hierarchic structures and policies exist in OSCs, there is no incentive, such as money in companies, which adequately explains why these structures and policies are accepted. As a conclusion of the preceding discussion, it can be deduced that there is no singular dominating motivational aspect responsible for contributing work. Instead, there are many intrinsic and extrinsic factors acting in concert.

3.2 Specific Open Source Control Instruments

The adequacy of governance tools is related to which motivational factors characterize individuals. Conventional control mechanisms are not usable in systems based on volunteer work. There is no possibility to penalize or reward members financially.

The efficiency of implemented governance instruments is linked to a large extent to the motivational basis of the involved individuals. Furthermore, motivation is related to different member groups and life cycle stages. That is why an expedient enlargement of governance tools must account for the motivational basis of all member groups.

On the other hand, neglecting to implement further coordination tools leads to losses in efficiency and growth. Divergent objectives probably cause stagnation or the decline of the whole project because of increasing information and transaction costs connected with a growing number of members [1, 31]. But implementing inappropriate control mechanisms or too many of them may lead to a decreased feeling of self-determination, which causes people to leave the project [67].

Reduction of asymmetric information, a natural phenomenon in growing organizations, and increasing transparency are essential objectives of control and monitoring mechanisms. Using highly developed technological communication tools, e.g. chat rooms, mailing lists, and CVSs, asymmetric information can be avoided. Additionally, these tools support transparency within the OSCs, help to increase trust between members, and lay open their motivational situation.

3.3 Social Control

Governance instruments are implemented for controlling, supervision and monitoring purposes as well as to guarantee the transparency in micro and macro structures.

Like conventional companies, open source projects must adjust their governance tools as increasing

complexity and the number of members increase. Basically, three forms of governance are applicable [30]:

- direct governance - inspection of behavior (behavioral control), e.g. on the basis of standards derived from experience [38];
- indirect governance - determination of output based on given goals (output control) [63, 38];
- social governance (social control) - comparison of conformity to certain morals and cultural rules [48].

In organization theory, traditional forms of governance (behavioral and output control) are examined in detail and mostly uniformly. The direct and indirect governance can hardly be applied in open source communities. Even if direct and indirect monitoring may be possible, direct and indirect control is hardly feasible. However, with the establishment of network-like organizational structures, as open source communities, the relevance of concepts of social control increases.

In particular, social control is based on the concept of trust, which is defined as the voluntary input of risky assets under the absence of explicit contractual protection and control [30].

Trust becomes a necessary prerequisite to assure the existence of flexible organizational structures [14], which is why it is identified as a key factor for the successful and growth and operation of an open source project. Additionally, trust forms the basis for the successful configuration and operation of open source communities [see 14] and is considered as a constructional attribute of all forms of virtual and distributed organizations [see e.g. 51, 62]. Furthermore, trust as a concept should guarantee that the partners are able to manage and organize processes at least partly independently [6]. In the case of open source communities, members must rely on the assumption that their contributions are applied to the project.

Instruments for the practical application of social control can be identified particularly in relation to the level of objective and personnel management [63, 30].

Specialized social norms and frameworks are able to support the genesis of trust in and among organizations. Open source projects draft a kind of ethical code (e.g. "Bylaws of the Apache Software Foundation") in many cases. Such an ethical and behavioral guideline assures a common feeling of identification.

For the implementation of social control, different governance instruments can be introduced such as the activation of common cultures among net partners with homogeneous value concepts or the review and creation of similar moral concepts by rituals or ceremonies – like regular meetings of Linux community members in nearly all large cities [1]. In addition, rules as guidelines for

operational behavior [20], the intensive employment of modern and uniform information and communication tools [3, 25] are also fields of social control.

Participants of open source projects exhibit a variety of cultural and technical abilities.

On the basis of the theoretical remarks and of the practical examples specified in relevant investigations, it can be derived that with an increasing degree of virtualization the significance of social control in the governance process increases [30]. Particularly in the introduction phase and on the levels of the bug fixers and programmers, where the coordination structures are weaker, social control is of high importance.

Referencing virtual organizations which are in significant aspects similar to OSCs, the following main tasks for the management of OSCs can be determined [5, 57]:

1. *Selection*: The organizational units defining the project must be decided upon.
2. *Information and Communication*: The managers must organize and evaluate communication and the flow of information.
3. *(Re-)detection*: The management must define how to (re-)distribute tasks within the OSC.
4. *Trust*: A goal of the leaders in an open source project must be to create confidence on a long-term basis with regard to the organization.

On each management level (structural, objective, personnel, as defined by Thomson [63]), instruments of social control are identifiable, such as the examples mentioned before.

One of the main tasks of managers in open source organisations is to use these instruments to strengthen the feeling of community. Doing this requires to know about the importance and existence of social control.

In contrast to traditional enterprises, open source projects use social control mechanisms for the solution of coordination and motivation problems with different intensity. The established hierarchies and coordination mechanisms are to be regarded as secondary in relation to the factors of social control.

Governance instruments that are used as control tools should ensure constructive cooperation of community members. Furthermore, these tools should maintain the quality of software. Quality assurance can be achieved by using peer review or by threatening members with loss of reputation within the community if they are suspected of misconduct.

3.4 Governance Instruments Affected by Life Cycle Stages and Member Groups

As previously mentioned, there are different requirements for governance tools to guarantee coherence and

successful product development in specific life cycle stages (see tab. 2).

	Introduction	Growth	Maturity	Decline or Revival
Focus	Idea Generation	Expansion	Stability	Adaption
Structure	Completely informal	More formal, centralized	Somewhat formal, decentralized	Slightly formal but less adherence
Division of Labor	Generalists	Some specification	Highly specialized	Less specialized
Coordination	Informal, one-on-one	Technology introduction	Formal, technology - intensive	Formal but less adherence
Examples⁴	Dam, HTMLarena plus accessibility	Eclipse, Typo3	Linux, Apache, Mozilla	Gnutella

Table 2: Life cycle stages in OSCs [see 67]

Therefore, the character and timing of instrument implementation are most important. Existing motivations must be supported in each member group and life cycle stage to guarantee the continuity of the project.

Introduction: The first step in developing open source software is to initiate the project by producing a working version of the software and to distribute the vision to the community [58, 66]. In this stage, the core group consists of just a few members. Project success is highly dependent on the quality of the initial idea, which will attract more programmers to participate and helps to reach a critical mass of developers. As long as no usable software exists, bug fixers will not support the project by messaging errors or insufficiencies. The organizational structure is quite informal and dominated by one-on-one communication and trust, so there is no need for an active management of resources [67].

Motives, like fun of self-determined programming, are of intrinsic nature in this stage.

Growth: Due to the increasing number of members, there is a growing need for (traditional) organizational structures such as hierarchies and communication rules. The growth stage is further characterized by a higher degree of specialization and is dependent on more coordination. Extrinsically motivated bug fixers join the project because there is a usable version of software and they want to highlight errors to increase efficiency.

Founders' activities change since they perform mainly management tasks instead of programming. For example in the Apache Project managers for specific subprojects or treasury are implemented [see <http://apache.org>]. Incentives dedicated to increasing the extrinsic motivation then gain importance to compensate for lower intrinsic motivation.

⁴ The examples are categorized by the number of downloads. The data originates from <http://sourceforge.net>.

Governance instruments should ensure that available organizational resources are managed efficiently. Transaction costs and information costs should be minimized by these tools.

Communication tools help to discuss further software development and optimization, support knowledge exchange (which satisfies demands for learning and teaching), and improve a feeling of community [19, 56]. Users of open source software receive technical support through communication forums, leading to accelerated growth.

Governance tools must also retain the motivation of community members. By publishing lists of most active programmers and bug fixers, e.g. at <http://sourceforge.net>, the incentive of increasing reputation is addressed. By choosing explicit entry requirements for these lists, managers gain a control instrument. However, these effects are not very strong because they do not affect intrinsic motivation factors. Moreover, too many rules and policies may reduce intrinsic motivation. Unlike managers, programmers will not compensate for a loss of intrinsic motivation with higher extrinsic motivation.

Maturity: In this stage, the number of members and downloads are at their maximum. The entire project is divided into numerous modules and cooperating programs exhibited by a high degree of specialization, e.g. the Apache project presently includes about twenty different software projects (see www.apache.org). For this reason, more coordination mechanisms must be implemented.

Activities of managers no longer include programming; instead, they must coordinate the entire organization, control software development, and communicate with cooperative companies (e.g. Apache and IBM, HP) or other projects. Public relations instruments become more important and make it possible to influence the reputation of the whole project. Managers are dominated by extrinsic motivation like career planning, as programmers are still intrinsically and extrinsically motivated.

In this stage, governance tools should support a feeling of community for all members, making a high degree of transparency within all aspects of the organization. A loss of transparency may lead to decreased community identification and lower motivation. One instrument to avoid this development, applied in practice, is to publish a codex or constitution which must be accepted by all community members and contains organizational and ethical policies (e.g. Apache Bylaws: <http://apache.org>, or Linux membership rules: <http://www.linuxquestions.org>). However, this rule should consider and perhaps support self-determination of all members.

Decline or Revival: After passing the maturity phase the project may enter a decline and an optional revival stage. The decline stage is dominated by a decreasing number of community members and downloads.

Reasons for decline are various. One may be inappropriate governance tools such as the following examples:

- Self-determination of programming is constrained by too many rules.
- Manager decisions are not understood by other members, causing a loss of transparency.
- Too much influence of profit-oriented companies within the open source project which undermines a feeling of community.
- Founders or important managers may leave the project. The community loses knowledge and identifying personalities (like Linus Torvalds is in the Linux project).

If these drivers are not stopped, open source projects may decline because of a loss of intrinsic and extrinsic motivation. Wynn concludes that there is a possibility of a project revival in some cases [67]. Revival can be successful if governance instruments are implemented and respectively readjusted in an appropriate manner to the basic demands of the community, which means considering life cycle stage and the nature of the member.

4 Conclusion

As this contribution shows, even open source projects must handle organizational questions, due to differences in goals and motives of the participation in such nonprofit organizations. The application of management instruments must consider intrinsic and extrinsic motivational aspects, even more than in profit-oriented companies. Social control mechanisms like trust, moral concepts by rituals or ceremonies, or group evaluation processes, like peer reviews among community members become a crucial part in the governance of open source communities because traditional monetary bonuses or sanctions are not working. Moreover, situation-based and adequate application of governance tools must be implemented.

No systematic approaches for the governance of open source projects have yet been found in practice. This contribution provides a nucleus for further research to derive approaches for adequate governance in OSCs. Therefore, best practice examples will be identified and matched within the proposed framework based on the properties of different member groups and concerns regarding the project life cycles. Further empirical analysis may show details of motivational aspects of each member group (bug fixer, programmer, and manager) and in different life cycle stages (introduction, growth, maturity, and decline).

5 References

- [1] Achtenhagen, L., J. Müller-Lietzkow, D. zu Knyphausen-Aufseß (2003), "Das Open Source-Dilemma: Open Source-Software zwischen freier Verfügbarkeit und

Kommerzialisierung", *Schmalenbachs Zeitung für betriebswirtschaftliche Forschung*, Düsseldorf, pp. 455-481.

- [2] Ahuja, M. K., K. M. Carley (1999), "Network Structure in Virtual Organizations", *Organisation Science: A Journal of the Institute of Management Sciences* **6** (10), pp. 741 - 758.
- [3] Albers, S., D. Bisping, K. Teichmann, J. Wolf (2002), "Management virtueller Unternehmen", Forschungsbericht.
- [4] Apache Software Foundation (1999), "Bylaws of the Apache Software Foundation", (Accessed: May 28, 2004): <http://www.apache.org/foundation/bylaws.html>.
- [5] Arnold, O., W. Faisst, M. Härtling, P. Sieber (1995), "Virtuelle Unternehmen als Unternehmenstyp der Zukunft?", *Handbuch der modernen Datenverarbeitung* **32** (185), pp. 8-32.
- [6] Behrens, S. (2000), "Produktionstheoretische Perspektiven der Virtuellen Unternehmung", *Zeitschrift für Betriebswirtschaft* **70** (2): "Virtuelle Unternehmen", pp. 157-175.
- [7] Benkler, Y. (2002), "Coase's Penguin, or, Linux and the Nature of the Firm", *The Yale Law* **112** (3).
- [8] Bezroukov, N. (1999), "Open Source Software Development as a Social Type of Academic Research", *First Monday* **4** (10), Chicago.
- [9] Bradach, J., R. Eccles, R. Price (1989), "Authority, and Trust: From Ideal Types to Plural Forms", *Annual Review of Sociology* **15**, pp. 97-118.
- [10] Daily, Catherine M., Dalton, Dan R., Cannella Jr., Albert A. (2003), "Corporate Governance: Decades of Dialogue and Data", *Academy of Management Review* **28** (3), pp. 371-382.
- [11] Deci, E.L., R.M. Ryan (2000), "The "What" and the "Why" of Goal Pursuits: Human Needs and the Self-Determination of Behavior", *Psychological Inquire* **11** (4), pp. 227-268.
- [12] Dempsey, B.J., D. Weiss, P. Jones, J. Greenberg (1999), "A Quantitative Profile of a Community of Open Source Linux Developers", <http://www.ibiblio.org>.
- [13] Donaldson, L., J.H. Davis (1991), "Stewardship theory or agency theory: CEO governance and shareholder returns", *Australian Journal of Management* **16**, pp. 49-64.
- [14] Ebner, W., Leimeister, J.M., Kremer, H. (2003), "Vertrauen in virtuellen Communities", *Uhr, W., Esswein, W., Schoop, E. Wirtschaftsinformatik 2003, Band II – Medien Märkte Mobilität*, Physika: Wiesbaden, pp. 619-628.
- [15] Eberl, P. (2002), "Vertrauen oder Kontrolle im Unternehmen?", Kahle, E. (ed.), *Organisatorische Veränderungen und Corporate Governance*, Deutscher Universitätsverlag: Wiesbaden.
- [16] Feller, J., B. Fitzgerald (2001), "Understanding Open Source Software Development", London: Addison-Wesley.
- [17] Frey, B.S. (1997), "Not just for the Money: An Economic Theory of Personal Motivation", Cheltenham.
- [18] Funkhouser, G. R., E. F. Shaw (1991), "How synthetic experience shapes social reality", *Journal of Communication* **2**.
- [19] Hermetsberger, A., C. Reinhardt (2004), "Sharing and Creating Knowledge in Open-Source Communities – The

- case of KDE”, Paper for Fifth European Conference on Organizational Knowledge, Learning, and Capabilities, Innsbruck.
- [20] Hertel, G., S. Niedner, S. Herrmann (2003), “Motivation of software developers in open source projects: An internet-based survey of contributors to the Linux kernel”, *Research Policy* **32**, pp. 1159-1169.
- [21] Hippel, E. von (2001), “Innovation by User Communities: Learning from Open Source Software”, *Sloan Management Review* **42** (4), pp. 82-86.
- [22] Jungwirth, C., E. Franck (2002), “Open versus Closed Software – Eine organisationsökonomische Betrachtung zum Wettbewerb der Betriebssysteme Windows und Linux”, Universität Zürich.
- [23] Kiesler, S., L. Sproull (1986), “Response effects in the electronic survey”, *Public Opinion Quarterly* **3**.
- [24] Kollock, P., M. Smith (1996), “Managing the Virtual Commons: Cooperation and Conflict in Computer Communities“, Herring, S. (ed.), *Computer-mediated Communication: Linguistic, Social, and Cross-cultural Perspectives*, Amsterdam, pp. 109-128.
- [25] Köhler, T. (2003), “Selbst im Netz? Die Konstruktion des Selbst unter den Bedingungen computervermittelter Kommunikation”, Westdeutscher Verlag: Opladen.
- [26] Kreps, D.M. (1997), “Intrinsic Motivation and Extrinsic Incentives”, *American Economy Review* **87** (2), pp. 359-364.
- [27] Krogh, G. von, S. Spaeth, K. Lahani (2003), “Community, Joining and Specialization in Open Source Software Innovation: A Case Study“, *Research Policy* **32** (7), pp. 1217-1241.
- [28] Lakhani, K.R., E. von Hippel (2001), “How Open Source Software Works: “free” user-to-user assistance”, *Research Policy* **32**, Cambridge.
- [29] Lamberti, H.-J. (2003), “Open Source – eine Alternative zu kommerziell lizenzierter Software?”, *Wirtschaftsinformatik* **45** (4), pp. 474 – 482.
- [30] Lattemann, C., T. Köhler (2004), “Vertrauen ist gut – Kontrolle ist besser? Ein Governance-Konzept für virtuelle Unternehmen.“, *Multikonferenz Wirtschaftsinformatik I*, Infix, pp. 306-323.
- [31] Lawrence, P.R., J.W. Lorsch (1967), “Organization and Environment: Managing Differentiation and Inegration”, Boston: Harvard University Press.
- [32] Lerner, J., Tirole, J. (2000): “The Simple Economics of Open Source”, *The National Bureau of Economic Research*, Inc. (Accessed: April 4, 2001): <http://papers.nber.org/papers/W7600>.
- [33] Lerner, J., Tirole, J. (2002), “The Scope of Open Source Licensing”, *Working Paper Harvard Business School*, Boston, M.A.
- [34] Link, J., N. Gerth, E. Voßbeck (2000), “Marketing Controlling”, München, Vahlen Verlag.
- [35] Mahoney, S. (2003): “Non-Profit Foundations and their Role, in Community-Firm Software Collaboration”, *Proceedings of the HBS - MIT Sloan Free/Open Source Software Conference 2003*.
- [36] Malone, M. S., W. H. Davidow (1992), “The Virtual Corporation. Structuring and Revitalizing the Corporation for the 21st Century”, Harper Collins: New York.
- [37] Magretta, J. (1998), “The Power of Virtual Integration: An Interview with Dell Computer’s Michael Dell”, *Harvard Business Review* **76** (2), pp. 73-84.
- [38] Merchant, K. (1985), “Control in Business Organizations”, Pitman: Boston.
- [39] Miles, R., W. Creed (1995), “Organizational Forms and Managerial Philosophies. A Descriptive and Analytical Review”, Staw, B. Cumminge, L. (eds.), *Research in Organizational Behavior* **17**. Greenwich (CT), JAI, pp. 333-372.
- [40] Miles, R. E., C. C. Snow (1986), “Organizations: New concepts for new forms”, *California Management Review* **28** (3), pp. 62-73.
- [41] Moon, J.Y., L. Sproull (2000), “Essence of Distributed Work: The Case of the Linux Kernel”, *First Monday* **5** (11).
- [42] Newby, G.B., J. Greenberg, P. Jones, “Open Source Software Development and Lotka’s Law: Bibliometric Patterns in Programming”, *Journal of the American Society for Information Science and Technology* **54** (1), New York.
- [43] Okkonen, J. (2002), “Performance in virtual organisations”, *Frontiers of e-Business Research* **1**, pp. 267-279.
- [44] Open Source Initiative (ed.) (2004), “The Open Source Definition”, (Accessed May 25, 2004): <http://www.opensource.org/docs/definition.php>.
- [45] O’Reilly, T. (2000), “Open Source: The Model for Collaboration in the Age of the Internet”, *Computers, Freedom and Privacy*, Toronto, Canada, Harvard University Press.
- [46] Osterloh, M., S. Rota, B. Kuster (2003), “Open Source Software Produktion: Ein neues Innovationsmodell?”, Forschungsarbeit am Institut für betriebswirtschaftliche Forschung der Universität Zürich, November 2003.
- [47] Osterloh, M., B.S. Frey (2000), “Motivation, Knowledge Transfer, and Organizational Forms“, *International Journal of the Economics of Business*, Special Issue on New Organizational Forms **9** (1), pp. 61-77.
- [48] Ouchi, W. (1979), “A Conceptual Framework for the Design of Organizational Control Mechanisms”, *Management Science* **25**, pp. 838-848.
- [49] Palmer, J.W., C. Speier (1997), “A Typology of Virtual Organisations: An Empirical Study”.
- [50] Postmes, T. (1997), “Social influence in computer-mediated groups”, Enschede: Print Partners Ipskamp.
- [51] Powell, W.W. (1996): “Trust-based forms of governance” Kramer, R.M., Taylor, T.R. (eds.) *Trust in organizations: Frontiers of strategy and research*, Thousand Oaks, CA: Sage, pp. 51-67.
- [52] Raymond, E. S. (1999), “The Magic Cauldron”, (Accessed: June 6, 2004): <http://www.catb.org/~esr/writings/magic-cauldron/>.
- [53] Robles, G. (2001), “WIDI – Who is doing it? Knowing more about developers”, *Informatik und Gesellschaft*, Institute of the Technical University of Berlin.
- [54] Robey, D. (1982), “Designing Organizations: A Macro Perspective”, Homewood, IL: Richard D. Irwin, Inc.
- [55] Schmitz, K., A. Ehrenforth (1996), Projekt D. Erlangen: Unveröffentlichte Handanweisung, Universität Erlangen-Nürnberg.

- [56] Schön, D. (1999), "The Reflective Practitioner – How Professionals Think in Action", New York, Basic Books.
- [57] Schreyögg, G. (2001), "Grundlagen moderner Organisationsgestaltung mit Fallstudien", Gabler, Wiesbaden.
- [58] Schweik, C.M., A. Semenov (2003), "The Institutional Design of Open Source Programming: Implications for Addressing Complex Public Policy and Management Problems", *First Monday* 8 (1), Chicago.
- [59] Shah, S. (2003), "Understanding the Nature of Participation & Coordination in Open and Gated Source Software Development Communities", *Proceedings of the HBS - MIT Sloan Free/Open Source Software Conference 2003*.
- [60] Snow, C., J. Lipnack, & J. Stamps (1999), "The virtual organization: promises and payoffs, large and small", Cooper, C. L. & Rousseau, D. M. (eds.), *The virtual organization*. Chichester: Wiley.
- [61] Sommergut, W. (2004), "Apache plant eigenen J2EE-Server", *computerwoche online*, (Accessed May 31, 2004):
<http://www.computerwoche.de/index.cfm?pageid=255&artid=52438&type=detail&category=21>.
- [62] Sydow, J. (1998), "Understanding the Constitution of Interorganizational Trust", Lane, C. Bachmann, R. (eds.), *Trust Within and Between Organisations. Conceptual Issues and Empirical Applications*. Oxford, Oxford University Press, pp. 31-63.
- [63] Thomson, J. *Organizations in Action*, New York, 1967.
- [64] Torvalds, L. (1998), "What motivates free software developers?", *First Monday* 3 (3).
- [65] Ye, Y., K. Kishida, (2003), "Toward an understanding of the motivation Open Source Software developers", *Proceedings of the 25th International Conference on Software Engineering*.
- [66] Ward, A. (2003), "The Leadership Lifecycle", Houndsmill, Basingstoke, Hampshire: Palgrave MacMillan.
- [67] Wynn, D.E. (2003), "Organizational Structure of Open Source Projects: A Life Cycle Approach", Abstract for 7th Annual Conference of the Southern Association for Information Systems, Georgia.