A Longitudinal, Multi-Project Study of Bug Tracking Productivity and Learning in Open Source Software Development

Janina Matz
Darmstadt University of Technology
Chair of Information Systems
Hochschulstr. 1,
64289 Darmstadt, Germany
matz@is.tu-darmstadt.de

Arne Beckhaus
SAP Research
Bleichstr. 8,
64283 Darmstadt, Germany
arne.beckhaus@sap.com

Dierk Erdmann
Darmstadt University of Technology
Chair of Information Systems
Hochschulstr. 1,
64289 Darmstadt, Germany
erdmann@is.tu-darmstadt.de

Peter Buxmann
Darmstadt University of Technology
Chair of Information Systems
Hochschulstr. 1,
64289 Darmstadt, Germany
buxmann@is.tu-darmstadt.de

ABSTRACT

The dynamics of Open Source Software development have recently received a lot of attention from an organizational learning perspective. Following a network theoretic approach, we study the temporal development of communication network structure and productivity in order to detect associations among these constructs. Thereby, we identify a research gap in that related work either focuses on too few projects or utilizes insufficient numbers of analyzed timeframes. Our study is both multi-project and longitudinal in order to detect holistic influencing factors of successful Open Source Software development. First, we find that learning effects are present since productivity increases over time. Next, we observe that growth in team size impedes productivity whereas the continuous concentration on central nodes coincides with increasing productivity. Against our expectation, we also find that increasing centralization does not yield decreasing network density, possibly due to Open Source Software developers deliberately avoiding the dependence on bottleneck nodes.

Keywords

Open Source Software, software development, organizational learning, productivity, network analysis

10^{th} International Conference on Wirtschaftsinformatik,
$16^{th} - 18^{th}$ February 2011, Zurich, Switzerland

1. INTRODUCTION

Research on Open Source Software communities has flourished as a consequence of scholars pointing to the vast potential of information that is coded in these platforms, (e.g., [?]). Thereby, various research streams can be differentiated. Researchers have been interested in questions regarding participation in Open Source Software communities and the motivational incentives for contributions therein [?, ?, ?, ?]. Closely aligned with this view is the question of how new members are attracted to a community [?]. It frequently takes a network perspective that is also addressed in the research streams of socialization [?] and social networks [?, ?]. In addition to the different explanatory goals, studies on Open Source Software are also conducted with differing focus. While a large stream of research rather takes a focused view by observing the bug fixing process [?, ?, ?], other works take a market perspective by analyzing the competition with proprietary software [?, ?, ?].

Within this broad field of research on Open Source Software development, this study's objective is to explain productivity from a network theoretic perspective. Thereby, our research intents to detect temporal effects in order to address the dynamics of many Open Source Software projects. It is hence in line with related work on organizational learning in the field of software engineering in general [?, ?] and Open Source Software development in particular [?]. As will be detailed in Section 4, we see a research gap in the design of this analysis. Related work has primarily chosen an unbalanced relation between the number of Open Source projects studied and the amount of temporal data available therein. We address this research gap by conducting a study that is both multi-project and longitudinal. With this research design, we intent to answer the research question whether network

theoretic constructs and productivity show corresponding temporal developments. In other words, we analyze, e.g., whether increasing communication network centrality coincides with increasing productivity. Due to studying these associations for different projects at the same time, we are able to holistically identify the dynamics of successful Open Source Software development.

The remainder of this paper is structured as follows. In Section 2, we present the theoretical foundation of our work. From this foundation, research hypotheses are derived in Section 3. Section 4 introduces our research design with a special focus on differentiation from related work. Section 5 explains the data source analyzed in this study. Suitable variables for the measurement of the postulated effects are presented in Section 6 and analyzed in Section 7. Potential implications of our results are discussed in the conclusions in Section 8.

2. THEORETICAL FOUNDATION

Research on Open Source Software communities is frequently based on two alternative theories. Crowston's coordination theory [**?**, **?**, **?**] is a suitable foundation for studies that analyze process flows. It builds on a rather strict formalization of the underlying process and hence does not optimally reflect the 'creative chaos' types of coordination in communities. This view is rather supported by the second frequently chosen theoretical basis being Granovetter's embeddedness theory [**?**,**?**,**?**]. Studies on Open Source Software communities that take this network embeddedness perspective usually analyze network structure and its association with performance (e.g., [**?**]). Our work follows this research stream and uses network embeddedness as its theoretical foundation. We hence believe that informal communication network structure is more important than formal coordination processes. This view is in line with theories of self-coordinating virtual communities that have appeared in various research disciplines such as virtual online groups [**?**], knowledge management [**?**], and information systems [**?**].

In addition to network embeddedness being the theoretical foundation of this article, it also significantly draws from organizational learning literature. This can primarily be attributed to the longitudinal research design that facilitates the analysis of learning effects over time. While researchers have stated that the term 'learning' represents different concepts in different research streams [**?**], organizational learning on an abstract level represents the need for firms and their employees to continuously learn and transfer knowledge [**?**]. Learning effects have consequently been studied and identified in various industries including software engineering (e.g., [**?**, **?**]). Thereby, Open Source Software development has been said to be "a particularly rich environment for studying organizational learning" [**?**, p. 485].

3. RESEARCH HYPOTHESES

It is common to many studies in the field of information systems research to analyze variances in order to map input factors to some kind of output measure. The definition of output is thereby discussed controversially. While for information systems success from a customer perspective, DeLone and McLean's [**?**] model has become the de-facto standard, software development is an information work process whose productivity is difficult to define [**?**, **?**]. Despite the challenges to define a productivity centric outcome measure, many studies in related work use this factor as a dependent variable. In the context of organizational learning, it plays an even more crucial role. The assumption that productivity increases over time is the essence of organizational learning theory (cf. [**?**]). Hence, we hypothesize that in Open Source Software projects, learning effects are present and that productivity and time are positively associated.

Hypothesis 1. *Productivity increases over time.*

Studies on organizational learning often only consider the phenomenon from a productivity and time perspective [**?**]. However, many other factors are of interest and can reveal interesting learning effects. The remaining hypotheses thereby do not directly focus the time construct but rather discuss the associations of other pairs of factors that are later on related to their temporal dynamics.

Mixed effects have been reported to exist in the association of productivity and team size. According to [**?**], this can be attributed to the opposed effects of large teams' coordination overheads and small teams' resource utilization challenges. The debate goes back to Brook's [**?**] seminal writing on the mythical man-month, in which he argues why doubling the man-power of a project might not increase but even decrease productivity due to the quadratical rise in potential communicative links. This effect has been found to also exist in the context of Open Source Software development [**?**].

From an economic perspective, the association of productivity and team size can be related to the debate on economies of scale in software engineering. Economies of scale are defined to exist in scenarios of increasing average productivity with rising volume [**?**] and have been frequently analyzed in the application context of software engineering (e.g., [**?**, **?**, **?**]). However, these studies lack a consensus on findings with studies reporting the full range of positive, negative, and not existing associations of productivity and size [**?**, **?**]. As already outlined in the previous section on the research design, many of these studies conduct a cross-sectional analysis of size and productivity levels across projects. The advantage of our study's approach to analyze multiple Open Source Software projects longitudinally is also evident in the analysis of economies of scale. In accordance with our theoretical foundation on network embeddedness theory, we postulate that the communication overhead of large teams rather impedes productivity.

Hypothesis 2. *Productivity and team size are negatively associated.*

The analysis of network structure and especially centrality has gained popularity over the last decades in information systems research and the social sciences [**?**,**?**]. Network centrality has also been identified to be an influential concept in the research domain of Open Source Software development [**?**,**?**]. In fact, the onion-like, core/periphery communication structures of a small, highly central group of developers and

a large, decentral group of users has been named an identifying characteristic of the Open Source Software development model [?, ?, ?]. While many studies find centrality to be beneficial for performance (e.g., [?]), partly due to efficiency in information flow, others report examples where too much centrality that ultimately takes the form of separation also impedes performance in Open Source Software development [?]. Due to our study's focus on individual project, we hypothesize that centrality is beneficial for development productivity since it resembles the Open Source Software movements core/periphery characteristic. If the focus would have been on larger collaboration networks of active developers (as is the case in many studies analyzing large parts of the Sourceforge.net community, e.g., [?, ?]), the opposite effect might have been more predominant.

Hypothesis 3. *Productivity and communication network centrality are positively associated.*

The fourth research hypothesis regards typical effects of network theory and consists of two parts. At first, we postulate that communication becomes more difficult when many nodes are present in the communication network. As Wu and Goh state, "a higher project density makes the dissemination of knowledge more time-consuming, as information and knowledge needs to travel through the extended hierarchies of the project team."[?, p. 4] Since density reflects the number of present pairwise communicative links in relation to all theoretically possible links in a network of the same size, increases in size require quadratic increases in new communication links in order for constant density. Hence, we hypothesize that communication network density and team size are negatively associated as it is very difficult for large teams to be as connected as small ones.

Hypothesis 4a. *Team size and communication network density are negatively associated.*

The second part of Hypothesis 4 also regards network density. We hypothesize that networks with high centrality require less communication links among peers due to the existence of key players. Information is efficiently routed over these central nodes and information paths become shorter [?]. As a consequence, we assume that network density can be low for these highly central networks since less communicative links are required when a lot of pairwise communication is routed over central nodes.

Hypothesis 4b. *Communication network density and centrality are negatively associated.*

4. RESEARCH DESIGN

The research designs of related work primarily differ in two dimensions. First, different data sources are chosen. Some studies make use of surveys among developers (e.g., [?]). Data acquisition by means of questionnaires has the advantage to retrieve diverse information ranging from age, gender, cultural background to experience. It is, however, also very limited in size. The alternative data acquisition strategy is to access existing archival data sources (e.g., [?]). Software engineering is a digitized process with large amounts of information available in databases. While less information can be retrieved per individual in comparison with surveys, archival data sources have the advantage that they are nearly unlimited in size and structural properties of networks are often only visible at this scale [?]. We hence conduct our research on the basis of archival data such as bug tracking systems and mailing list archives.

The second important dimension of research design regards the temporal nature of the study. Many works analyze static phenomena in cross-sectional studies (e.g., [?]). While this design has the advantage to thoroughly acquire data and measure effects, it neglects the dynamics of the fast moving software engineering domain. Hence, a second stream of research conducts longitudinal studies that explicitly analyze temporal development of effects (e.g., [?]). We follow this stream and analyze the dynamics of Open Source Software development teams.

Several studies in related work can be identified that share our research design. However, our work differs from these studies in the implementation of the longitudinal, multi-project study. Some studies shift the focus towards rich information regarding the time factor. However, this comes at the expense of analyzing a reasonably large number of different projects. The works of [?, ?, ?] fall into this category. Other studies shift their focus in the opposite direction and analyze many projects at the expense of very limited data on timing. Often, factors are measured for a large number of projects at two points in time and a delta-study of these two measurements is conducted in order to reveal temporal effects (e.g., [?]). Our research design aims at providing a sufficient number of measurements for both projects and time slots. This well-balanced selection of projects and time slots comes at the expense of applicable research methodology. Just like the analysis of [?], our sample does not allow for standard time-series analysis. As will be elaborated on in Section 7, it is, however, suitable for correlation analysis and can reveal other and potentially more interesting findings than one of the two other research designs discussed above.

5. DATA SOURCE

For our analysis of communication patterns and Open Source Software project's productivity on a reasonable number of projects over time, there are several constraints on data acquisition. The biggest challenge is to identify projects that fulfill the criteria of consistent data over time. The analysis of communication network structure requires rather big projects in order to be confronted with interesting network effects that can reasonably be analyzed by means of standard social network analysis techniques. While large parts of related work on Open Source Software development use a large number of Sourceforge.net projects, we could not solely rely on this vast data source since it does not host very many big projects that are suited to our analysis of communication structures over time. We hence aggregated candidates from various Open Source Software directories and manually identified projects that provide sufficient amounts of data. Thereby, automated approaches such as that of [?] are not feasible in our setup since not all information is publically available. Instead, in some cases, we need to contact the projects' administrators and ask them for mailing

list archives. We do not believe that our data acquisition strategy introduces a serious bias since small Open Source projects, which were not covered by our study, do not often represent truly collaborative and already successful projects.

Thereby, we use the archived mailing lists of eleven different Open Source projects for analyzing communication patterns and structures. For a given period of time, a network graph is constructed, where each node represents a person. The graph's edges linking the persons represent the e-mails that have been exchanged. The edges' weights indicate the number of e-mails that have been exchanged between the corresponding two participants. Formally, a graph or network is given by $G = (\mathcal{V}_G, \mathcal{E}_G)$, where $\mathcal{V}_G$ is the set of vertices and $\mathcal{E}_G$ is the set of edges that constitute the graph. We do not consider directed networks that show the direction of the sent e-mails, as this is not relevant to our analysis.

For measuring productivity, we use the Open Source Software projects' bug tracking databases, which is in line with related work [**?**]. In the case of the analyzed projects, these additional data sources are freely accessible. They represent a type of workflow management system and help the developers to organize the bug fixing process. For each listed bug, the system keeps track of several attributes, e.g.,

- the initial reporting date,
- a description of the bug,
- its current status (e.g., 'fixed', 'open', ...), and
- the dates of subsequent actions, such as the resolution of the bug.

Using these attributes, we can define a measure of productivity as given in Section 6.1. Of course, the software development process and its productivity consists of much more than bug tracking. However, as elaborated on in the following, productivity of software engineering is very difficult to measure. We hence follow the argument of [**?**] that bug tracking is the one sub-process of software engineering that involves everybody of the community. Quality management, which bug tracking belongs to, is also a very important aspect of the entire software engineering process from a financial point of view that accounts for large parts of total software budgets [**?**].

6. VARIABLE MEASUREMENT

6.1 Productivity

While in traditional research settings, productivity has often been defined as 'output per input', in knowledge work, a definition of productivity as a combination of effectiveness and efficiency is becoming more common [**?**]. Effectiveness means 'doing the right things' and is in our setting of less relevance than efficiency ('doing things the right way') since software engineering tasks are rather fixed and well defined. Efficiency is the more interesting factor here as it can be seen as the main outcome measure influencing productivity.

An important argument for the higher importance of output than input in knowledge work productivity is data availability and measurement. While in conveyor-belt-type of work, input can be easily measured via operation hours, knowledge work is much more complex and interweaved over tasks. It is hence hardly feasible to precisely measure input for a given task. A knowledge worker usually executes diverse tasks in parallel and does not keep time on a fine grained level, such as minutes or even seconds. This is one of the reasons for the dominance of efficiency-based over output/input-based productivity definitions in knowledge work contexts.

In line with this knowledge work perspective on productivity, we follow Kidane and Gloor's [**?**] productivity definition in a software engineering and bug tracking context for a given period of time as

$$\text{productivity} = \frac{\text{Number of bugs reported } \textit{and} \text{ fixed}}{\text{Number of reported bugs}}.$$

This definition deviates from [**?**, p. 20] in that we only count bugs that have been reported *and* fixed in the same period of time. This is due to a correction of an apparent data inconsistency in the analyzed Open Source Software projects' bug tracking databases. These list large amounts of bugs as fixed on the exact same point of time with an accuracy of a second. Obviously, the "date of last change" does not represent the true bug resolution time in these cases but is rather the result of an automatic data correction procedure. However, the initial reporting date of the bugs did not exhibit any inconsistencies. If we included all these bugs with wrong timestamps, the observed productivity values would be much too high for certain periods of time. Our way of measuring productivity avoids this problem and is more robust against bug database entries with erroneous date entries.

6.2 Network Centralization

The second measure of network topology is centralization. We use betweenness centrality for measuring how centralized the communication networks are. While there exist other concepts of measuring centrality, the betweenness centrality is one of the most commonly used ones [**?**, **?**, **?**]. The betweenness centrality for a *single node* v is defined as

$$\mathrm{BC}(v) = \underbrace{\left(\sum_{s \neq v \neq t} \frac{\sigma_{st}(v)}{\sigma_{st}} \right)}_{\text{centrality}} \cdot \underbrace{\left(\frac{2}{(|\mathcal{V}_G| - 1)(|\mathcal{V}_G| - 2)} \right)}_{\text{normalizing factor}}, \quad (1)$$

where σ_{st} is the number of shortest paths from node s to t, and $\sigma_{st}(v)$ is the number of shortest paths between s and t that pass through node v. The "normalizing" factor in Equation (1) rescales the betweenness centrality to the interval $[0, 1]$ [**?**].

The centralization for a *network* G with n nodes is defined as

$$\mathrm{BC}(G) = \frac{\sum_{i=1}^{n} (\mathrm{BC}(n^*) - \mathrm{BC}(v_i))}{n - 1}, \quad (2)$$

where $\mathrm{BC}(n^*)$ is the maximum betweenness centrality value of a node in the whole network. The denominator $(n - 1)$ normalizes the entire network's centralization to a value between zero and one [**?**].

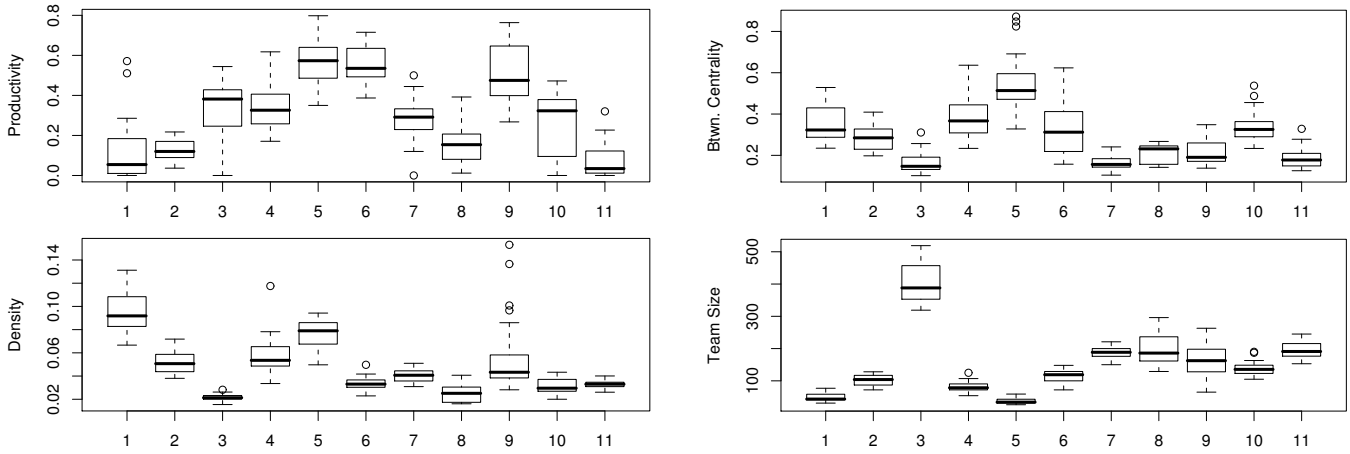

Figure 1: Overview of Intra-Project Variances.

6.3 Network Density

The third measure of network topology is density. It is defined as the number of edges present in the graph, divided by the maximum number of edges that could possibly exist. Therefore the network's density $\Delta(G)$ is given by [?, p. 129]

$$\Delta(G) = \frac{2 \cdot |\mathcal{E}_G|}{|\mathcal{V}_G| \cdot (|\mathcal{V}_G| - 1)}. \quad (3)$$

The density $\Delta(G)$ measures how connected different persons (nodes) of a network are. The networks we analyzed show rather small density values, usually below 0.1, which means that 10% or less of the possible connections between the nodes are actually established by mail exchange. However, small values for the density are not uncommon, as sociology regards values above 0.4 as high [?]. Studies have shown that networks with higher densities display a higher degree of mutual control and solidarity [?, p. 84]. Furthermore dense networks are said to represent communities that a more capable of finding and pursuing common goals than less dense ones. Anyhow, one should be careful to conclude that low density networks – like the ones we analyze here – exhibit a slow distribution of information (c.f. [?]), because findings for non-electronic social networks are only partly applicable to virtual communities [?]. Actually, researchers have shown that Open Source Software communities show rather fast patterns of information exchange [?].

6.4 Team Size

Our hypotheses of Section 3 include a size construct which we measure by the "team size" variable. In order to measure how many people have been participating in the communication during a certain period of time, we simply interpret the number of nodes in the communication network G as the size of the team. As defined in Section 5, every participant of the mailing list communication is represented by a single node in the network. Formally, the team size is given by the set of nodes' cardinal number:

$$\text{team size} = |\mathcal{V}_G|.$$

7. DATA ANALYSIS

Building on the variable definitions of the previous section, we now present the results of our data analysis. As stated in Section 4, our research design explicitly balances the amount of projects and timeframes available therein. The advantage of this holistic view on the dynamics of Open Source Software development come at the expense of applicable research methodology. In line with [?], we cannot apply standard time-series analysis in this setup but introduce an alternate methodological approach to answering our research question before presenting analysis results.

In this section, we present our approach for analyzing and interpreting the data sources described above. We first outline the aggregation of variables and measures and then explicitly address the problem of different variances across the OSS projects. This problem has been ignored by studies conducted before [?, ?].

7.1 Methodology

Our research methodology consists of three steps:

1. Analyze differences in variances across projects in order to demonstrate the necessity to analyze timeframes of different projects separately.
2. Per project, analyze the correlations of the variables with time.
3. Compare the results of the previous step across projects and legitimize the finding's statistical validity with an additional correlation and Pearson product momentum test.

In step 1, we explicitly address a shortcoming of parts of related work (e.g., [?, ?]) in that we analyze whether two randomly chosen samples of a single project are more alike than two randomly chosen samples of different projects. [?, ?] do not analyze these variances but choose a research design that inflates the number of samples by analyzing all timeframes of all projects in a single analysis. However, our methodology checks for the assumption that these samples are drawn from a unique distribution.

In step 2, we assume (and confirm in Section 7.2) that the variances among samples within a project are less than those

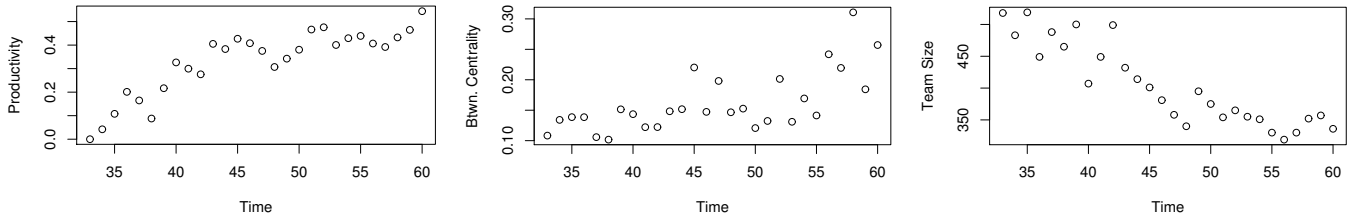

Figure 2: Project 'gcc'.

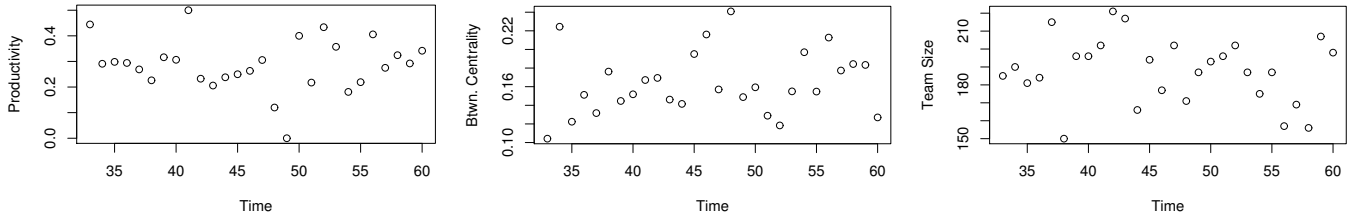

Figure 3: Project 'kde'.

spread over all projects and that a joint analysis of all available timeframes of all projects is therefore not methodologically sound. Instead, we rather analyze each project separately according to the following formal scheme:

For each project $p \in P$ and each variable v, let V_p be the set of corresponding measurements of v for the time slots analyzed for p. Further let

$$a_v = \frac{1}{|P|} \sum_{p \in P} \left(\frac{1}{|V_p|} \sum_{v \in V_p} v \right).$$

Then, for each project p, the first aggregated measure per variable v is

$$d_{v,p} = -a_v + \frac{1}{|V_p|} \sum_{x \in V_p} x.$$

It represents the difference of a project's average variable value from the cross-project average of the same variable. Hence, this measure does not yet reflect the longitudinal nature of our study but rather serves as a comparison to related work on a cross-sectional basis.

The time-dependent aggregation builds on significant correlations of a variable v with the time variable. For each project $p \in P$ and each variable v with measurements V_p and the time slots T_p, let $c_{v,p}$ be the Pearson correlation coefficient of the correlation between V_p and T_p if the Pearson product momentum test of this correlation is significant at the 5% level and 0 otherwise.

In step 3 of our research methodology, we compare the values of $d_{v,p}$ and $c_{v,p}$ across all analyzed projects. This comparison can reveal interesting findings on a qualitative level already. It can be observed which variables' temporal developments coincide. In addition to answering our research hypotheses for each project separately, this final step enables a holistic answer. In order to demonstrate the statistical significance of these findings, the associations of postulated effects are analyzed by correlating the $d_{v,p}$ and $c_{v,p}$ values of all eleven projects. Despite the small number of samples in this additional meta-analysis' correlation, the hypotheses' significance can be confirmed by means of the Pearson product momentum test.

7.2 Analysis and Results

The first step of analysis according to our research methodology as described in the previous section is to compare the variances of our variables across different projects. In line with related work, we choose intra-project variance boxplots for this analysis (e.g., [**?**]). Figure 1 shows four different charts, one for each variable. All x-axes show the eleven different projects and the y-axis shows the corresponding variable measure in each case. The boxplots in Figure 1 reveal that all four variables take very different levels across projects. There is little overlap between different boxplots of a subchart, which indicates that the measurements per timeslots of different projects must be seen to be drawn from different distributions. This result legitimizes our research design to analyze projects and timeslots separately. It must also be seen as a critique of part of related work that ignores the different variations (e.g., [**?**, **?**]).

Next, we take a more detailed look at the temporal development of our variables in two example projects. In Figure 2, three variables are plotted on the y-axes against time on the x-axis. The three scatterplots in this figure all represent the 'gcc' project. All three plots take a more or less linear form. Accordingly, the correlations of all three variables with time are significant at the 5% level. The opposite case is present in the project 'kde' which is depicted in Figure 3. The seemingly random plots yield insignificant correlations of the three variables and time. Hence, various of the effects postulated in our research hypotheses can be confirmed to exist in 'gcc' but cannot be observed in 'kde'.

The comparison of these two projects shows that the dynamics of Open Source Software development are complex and may vary a lot across different projects. Due to our longitudinal, multi-project research design, we are able to go beyond merely discussing the effects present in individual projects. We can rather derive holistic answers to our research hypotheses that consider all analyzed projects at

Project	Temporal Correlation				Delta from Avg. Level			
	Produc.	Centr.	Density	Size	Produc.	Centr.	Density	Size
ant	0.692	–	–	–	-0.173	0.070	0.047	-104
apache	–	–	0.480	-0.638	-0.181	-0.005	0.005	-50
gcc	0.865	0.670	0.555	-0.904	0.026	-0.128	-0.026	252
gimp	0.426	–	-0.630	0.680	0.034	0.094	0.011	-70
gnumeric	0.667	–	0.555	-0.683	0.260	0.266	0.029	-116
gtk	–	-0.606	–	–	0.260	0.042	-0.014	-37
kde	–	–	–	–	-0.017	-0.126	-0.007	35
maemo	0.771	–	0.887	-0.669	-0.141	-0.085	-0.022	51
python	-0.715	-0.402	-0.741	0.781	0.205	-0.076	0.007	9
samba	0.727	–	0.680	-0.708	-0.039	0.051	-0.016	-14
wine	0.747	–	–	–	-0.235	-0.103	-0.014	43

Table 1: Aggregated Correlation Results and Descriptive Statistics per Project.

the same time. For this purpose, the third step of our proposed research methodology consists of a meta-analysis of the temporal correlation results.

Table 1 serves as a basis for this analysis. It can be seen as an outcome of the project-centric analysis up to the second step of our research methodology. The projects are listed in the left column. For each project and each variable, two further values are given. The 2nd to 5th columns represent the Pearson correlation coefficients of the corresponding variables and time, i.e., $d_{v,p}$ as defined in Section 7.1. Where this correlation is not significant at the 5% level, a dash is displayed. For example, the three scatterplots in Figure 2 correspond with row 'gcc' and the columns 'Produc.', 'Size', and 'Centr.' in left part of Table 1.

In addition to the analysis of Open Source Software dynamics by means of temporal correlations, we also double-check our results by analyzing the static level of a variable in comparison to all projects. Table 1's four rightmost columns serve as a basis for this analysis. These columns represent the difference of the corresponding project's arithmetic mean and the cross-project arithmetic mean of the same variable, i.e., $c_{v,p}$ of Section 7.1. Analyzing these static levels provides valuable additional insights. For example, the size-delta of 252 for project 'gcc' reveals that despite this project's decreasing team size, 'gcc' is still much larger than the average of all analyzed projects (each averaged over the entire time covered by this study). Additionally, the right part of Table 1 representing the static levels of variables can be compared to the left part representing dynamics. For example, in project 'gnumeric', productivity and size show temporal correlations in opposite direction as postulated in Hypothesis 2. This effect can also be observed statically, since 'gnumeric' has an above average level of productivity but its size is below average. However, this comparison does not hold in many cases and does not yield statistical significance in the meta-analysis conducted below. It must hence be seen as an add-on to our analysis only.

Regarding Hypothesis 1, the second column (temporal correlation of productivity) reveals that seven projects show the expected association (embodied in the positive correlation coefficient), that project *python* exhibits the opposite direction, and that three projects do not have a significant association between productivity and time. The first hypothesis can hence be said to be confirmed and is statistically significant at the 5% level according to a t-test (the alternative hypothesis that the true mean productivity coefficient is negative can be rejected, $t = 2.543, p = 0.015$).

Hypothesis 2 postulates a negative association of productivity and team size. An additional correlation of the temporal variables for the corresponding two variables (productivity and size) yields a coefficient of -0.605 and is significant at the 5% level. Hence, projects whose size increases over time tend to show decreasing productivity and vice versa. This dynamic effect is also supported by the static average levels (right hand side of Table 1) which also yield a negative correlation coefficient, despite not being statistically significant. Those projects with comparably high levels of productivity are rather small in terms of team size and vice versa.

The third hypothesis is the first to regard network structure. It postulates a positive association of communication network centrality and productivity. The additional correlation of both corresponding dynamic variables yields a coefficient of 0.645 and is significant at the 5% level. Those projects that show increasing productivity rates also become more central over time and vice versa. Again, the effect is also visible when correlating the static variables, though lacking statistical significance.

The next hypothesis of network structure, Hypothesis 4a, regards a potential negative association of team size and communication network density. Its intuition is the larger teams get, the more difficult it is to communicate with everyone. The additional correlation of the temporal 'Size' and 'Density' variables yields a coefficient of -0.970 and is significant at the 1‰ level. The static variant confirms this drastic association and is also significant at the 1% level. Hence, both variables must be seen to be highly correlated both dynamically and statically.

The last hypothesis, Hypothesis 4b, also regards network density and postulates a negative association with network centrality. However, against our expectations, the opposite effect can be observed in our data. Despite not being statistically significant, the additional correlation reveals that centrality and density dynamics are coherent. Increasing

centrality seems to not necessarily lead to decreasing density and vice versa. A possible explanation is that projects want to benefit from shortened information paths through central nodes but also want to keep the less efficient communicative links as a backup and bottleneck prevention. When analyzing the static variant of this hypothesis, the same effect is present (coefficient 0.649) and is even significant at the 5% level.

8. CONCLUSIONS

In this study, we analyze the temporal development of factors that have previously been found to influence Open Source Software development success. Thereby, we identify a weakness of related longitudinal studies in this field. They either analyze very few projects for many timeslots or conduct a delta-study of two points in time for many projects. A third group of related work fails to account for significant variations between projects by analyzing timeslots of all projects jointly.

We address these shortcomings of related work by analyzing multiple projects over a reasonable number of timeslots. This research design comes at the expense of the unapplicability of standard time-series analysis techniques [**?**]. We apply a two-stage Pearson correlation-based research methodology which enables us to draw statistically significant findings.

In addition to the distinction from related work by our longitudinal, multi-project research design, we make an important contribution to the literature on collaboration in Open Source Software development with our findings. Our results show that the theory of organizational learning is applicable in the context of Open Source Software development. Most of the analyzed projects exhibit increasing levels of productivity over time. Our work hence confirms that the findings of learning theory in commercial software development (e.g., [**?**]) are also valid in the context of Open Source Software. The implications for project coordinators are manifold. For example, they should attempt to foster participants' long-term commitment to the project in order to fully exploit knowledge accumulation and learning benefits.

Our findings regarding diseconomies of scale and the negative temporal association of productivity and team size imply that projects should not become too big. Open Source Software project administrators should react to growth in size with strict modularization and splitting too big projects in parts in order to keep productivity high. In line with related work on collaboration network structure, we find that centrality is beneficial for productivity. It is thus desirable to identify and develop key contributors. Strengthening their position in the collaboration network yields rises in productivity.

A rather surprising finding of our work is the positive temporal association of collaboration network density and centrality. It implies that Open Source Software projects do not rely on central nodes too much in order to avoid bottlenecks. This collaboration network evolution suggests that the advantages of centrality – such as shortened paths of information flow – can be utilized, but that the risks associated with centralization – such as reliance on central nodes and the creation of potential bottlenecks – are prevented at the same time by maintaining backup collaborative ties. Should the central nodes become unavailable, work can quickly be routed via the less efficient, but nevertheless maintained backup connections. We are not aware of other works that have detected this efficient and failsafe communication strategy in the dynamics of Open Source Software development.

Future work should focus on transferring our results to an even larger number of Open Source Software projects. Additionally, more constructs than the analyzed ones can be transferred from the organizational learning and communication network literature. For example, researchers could transfer the results from related works studying membership dynamics to the context analyzed in this study.

9. REFERENCES

[1] P. Adams, A. Capiluppi, and C. Boldyreff. Coordination and productivity issues in free software: The role of Brooks' law. In *IEEE International Conference on Software Maintenance (ICSM 2009)*, pages 319–328, 2009.

[2] P. Anbalagan and M. Vouk. On mining data across software repositories. In *Proc. of the International Workshop on Mining Software Repositories (MSR 2009)*, pages 171–174, Los Alamitos, CA, USA, 2009. IEEE Computer Society.

[3] S. Aral, E. Brynjolfsson, and M. van Alstyne. Information, technology and information worker productivity: Task level evidence. In *Proc. of the International Conference on Information Systems (ICIS)*, Milwaukee, WI, 2006.

[4] L. Argote. *Organizational learning: Creating, retaining and transferring knowledge*. Kluwer Academic Publishers, Norwell, MA, 1999.

[5] Y. A. Au, D. Carpenter, X. Chen, and J. G. Clark. Virtual organizational learning in open source software development projects. *Information & Management*, 46(1):9–15, 2009.

[6] R. P. Bagozzi and U. M. Dholakia. Open source software user communities: A study of participation in Linux user groups. *Management Science*, 52(7):1099–1115, 2006.

[7] R. D. Banker and S. A. Slaughter. A field study of scale economies in software maintenance. *Management Science*, 43(12):1709–1725, 1997.

[8] D. Barbagallo, C. Francalenei, and F. Merlo. The impact of social networking on software design quality and development effort in open source projects. In *Proc. of the International Conference on Information Systems (ICIS)*, Paris, France, 2008.

[9] M. Barthélemy. Betweenness centrality in large complex networks. *The European Physical Journal B - Condensed Matter and Complex Systems*, 38(2):163–168, 3 2004.

[10] J. Bitzer and P. Schröder. The economics of open source software development: An introduction. In J. Bitzer and P. Schröder, editors, *The Economics of Open Source Software Development*, pages 1–13. Elsevier Science, 1 edition, 10 2006.

[11] W. F. Boh, S. A. Slaughter, and J. A. Espinosa. Learning from experience in software development: A multilevel analysis. *Management Science*, 53(8):1315–1331, 2007.

[12] S. P. Borgatti, A. Mehra, D. J. Brass, and G. Labianca. Network analysis in the social sciences. *Science*, 323(5916):892–895, Feb 2009.

[13] U. Brandes. On variants of shortest-path betweenness centrality and their generic computation. *Social Networks*, 30(2):136 – 145, 2008.

[14] F. P. Brooks. *The Mythical Man-Month.* Addison Wesley, New York, 1975.

[15] V. Buskens. *Social Networks and Trust (Theory and Decision Library).* Springer Netherlands, 1 edition, 3 2002.

[16] K. Crowston. A coordination theory approach to organizational process design. *Organization Science*, 8(2):157–175, Mar. - Apr. 1997.

[17] K. Crowston and J. Howison. Hierarchy and centralization in free and open source software team communications. *Knowledge, Technology and Policy*, 18(4):65–85, December 2006.

[18] K. Crowston, J. Howison, and H. Annabi. Information systems success in free and open source software development: Theory and measures. *Software Process–Improvement and Practice*, 11(2):123–148, 2006.

[19] K. Crowston and B. Scozzi. Bug fixing practices within free/libre open source software development teams. *Journal of Database Management*, 19(2):1–30, April-June 2008.

[20] K. Crowston, K. Wei, Q. Li, and J. Howison. Core and periphery in free/libre and open source software team communications. In *Proc. of the 39th Annual Hawaii International Conference on System Sciences (HICSS)*, Washington, DC, USA, 2006. IEEE Computer Society.

[21] W. H. DeLone and E. R. McLean. Information systems success: The quest for the dependent variable. *Information Systems Research*, 3(1):60–95, 1992.

[22] P. F. Drucker. Knowledge-worker productivity: The biggest challenge. *California Management Review*, 41(2):79–94, 1999.

[23] N. Ducheneaut. Socialization in an open source software community: A socio-technical analysis. *Computer Supported Cooperative Work*, 14(4):323–368, 2005.

[24] N. Economides and E. Katsamakas. Two-sided competition of proprietary vs. open source technology platforms and the implications for the software industry. *Management Science*, 52(7):1057–1071, 2006.

[25] M. Granovetter. Economic action and social structure: The problem of embeddedness. *American Journal of Sociology*, 91(3):481–510, Nov 1985.

[26] M. Granovetter. The impact of social structure on economic outcomes. *Journal of Economic Perspectives*, 19(1):33–50, 2005.

[27] M. S. Granovetter. The strength of weak ties. *American Journal of Sociology*, 78(6):1360–1380, May 1973.

[28] R. Grewal, G. L. Lilien, and G. Mallapragada. Location, location, location: How network embeddedness affects project success in open source systems. *Management Science*, 52(7):1043–1056, July 2006.

[29] J. Hahn, J. Y. Moon, and C. Zhang. Emergence of new project teams from open source software developer networks: Impact of prior collaboration ties. *Information Systems Research*, 19(3):369–391, 2008.

[30] D. Hu and J. L. Zhao. Discovering determinants of project participation in an open source social network. In *Proc. of the International Conference on Information Systems (ICIS)*, Phoenix, AZ, 2009.

[31] C. L. Huntley. Organizational learning in open-source software projects: An analysis of debugging data. *IEEE Transactions on Engineering Management*, 50(4):485–493, Nov 2003.

[32] J. Jaisingh, E. W. K. See-To, and K. Y. Tam. The impact of open source software on the strategic choices of firms developing proprietary software. *Journal of Management Information Systems*, 25(3):241–275, 2008.

[33] Y. Kidane and P. A. Gloor. Correlating temporal communication patterns of the eclipse open source community with performance and creativity. *Computational & Mathematical Organization Theory*, 13(1):17–27, 3 2007.

[34] B. A. Kitchenham. The question of scale economies in software—why cannot researchers agree? *Information and Software Technology*, 44(1):13–24, January 2002.

[35] J. Kleinberg. The convergence of social and technological networks. *Communications of the ACM*, 51(11):66–72, 2008.

[36] A. G. Koru and J. Tian. Defect handling in medium and large open source projects. *IEEE Software*, 21(4):54–61, July-Aug. 2004.

[37] Y. Long and K. Siau. Impacts of social network structure on knowledge sharing in open source software development teams. In *AMCIS 2008 Proceedings*, 2008.

[38] T. W. Malone and K. Crowston. What is coordination theory and how can it help design cooperative work systems? In *Proc. of the ACM Conference on Computer-Supported Cooperative Work (CSCW)*, pages 357–370, Los Angeles, CA, 1990.

[39] T. W. Malone and K. Crowston. The interdisciplinary study of coordination. *ACM Computing Surveys*, 26(1):87–120, 1994.

[40] M. McLure-Wasko and S. Faraj. Why should I share? Examining social capital and knowledge contribution in electronic networks of practice. *MIS Quarterly*, 29(1):35–57, 2005.

[41] J. Mewes. *Ungleiche Netzwerke? Vernetzte Ungleichheit: Persönliche Beziehungen im Kontext von Bildung und Status.* Vs Verlag, 1 edition, 3 2010.

[42] P. Middleton and J. Sutton. *Lean Software Strategies: Proven Techniques for Managers and Developers.* Productivity Press, New York, May 2005.

[43] W. Oh and S. Jeon. Membership herding and network stability in the open source community: The Ising perspective. *Management Science*, 53(7):1086–1101, 2007.

[44] E. Otte and R. Rousseau. Social network analysis: a powerful strategy, also for the information sciences. *Journal of Information Science*, 28(6):441–453, 2002.

[45] P. C. Pendharkar. Scale economies and production function estimation for object-oriented software component and source code documentation size. *European Journal of Operational Research*, 172(3):1040–1050, August 2006.

[46] R. D. Pritchard. *Productivity Measurement and Improvement: Organizational Case Studies*. Greenwood Publishing Group, 1995.

[47] N. Ramasubbu, S. Mithas, M. S. Krishnan, and C. F. Kemerer. Work dispersion, process-based learning, and offshore software development performance. *MIS Quarterly*, 32(2):437–458, 2008.

[48] E. S. Raymond. *The Cathedral and the Bazaar*. O'Reilly, Cambridge, MA, 1999.

[49] H. Rheingold. *The Virtual Community: Homesteading on the Electronic Frontier*. Addison-Wesley, Reading, MA, 1993.

[50] J. A. Roberts, I.-H. Hann, and S. A. Slaughter. Understanding the motivations, participation, and performance of open source software developers: A longitudinal study of the Apache projects. *Management Science*, 52(7):984–999, 2006.

[51] G. Robles, J. M. Gonzalez-Barahona, and I. Herraiz. Evolution of the core team of developers in libre software projects. In *Proc. of the International Workshop on Mining Software Repositories (MSR 2009)*, volume 0, pages 167–170, Los Alamitos, CA, USA, 2009. IEEE Computer Society.

[52] R. Sen. A strategic analysis of competition between open source and proprietary software. *Journal of Management Information Systems*, 24(1):233–257, 2007.

[53] A. Silberston. Economies of scale in theory and practice. *Economic Journal*, 82(325):369–391, 1972.

[54] K. J. Stewart, A. P. Ammeter, and L. M. Maruping. Impacts of license choice and organizational sponsorship on user interest and development activity in open source software projects. *Information Systems Research*, 17(2):126–144, 2006.

[55] Y. Tan, V. Mookerjee, and P. Singh. Social capital, structural holes and team composition: Collaborative networks of the open source software community. In *Proc. of the International Conference on Information Systems (ICIS)*, Montreal, Canada, 2007.

[56] G. von Krogh, S. Spaeth, K. R. Lakhani, and R. Policy. Community, joining, and specialization in open source software innovation: a case study. *Research Policy*, 32(7):1217–1241, July 2003.

[57] G. von Krogh and E. von Hippel. The promise of research on open source software. *Management Science*, 52(7):975–983, 2006.

[58] M. Wasko, R. Teigland, and S. Faraj. The provision of online public goods: Examining social structure in an electronic network of practice. *Decision Support Systems*, 47(3):254 – 265, 2009. Online Communities and Social Network.

[59] S. Wasserman and K. Faust. *Social Network Analysis: Methods and Applications*. Cambridge University Press, 11 1994.

[60] E. Wenger. *Communities of Practice: Learning, Meaning, and Identity*. Cambridge University Press, Cambridge, UK, 1998.

[61] J. M. Wilson, P. S. Goodman, and M. A. Cronin. Group learning. *Academy of Management Review*, 32(4):1041–1059, 2007.

[62] C.-G. Wu, J. H. Gerlach, and C. E. Young. An empirical analysis of open source software developers' motivations and continuance intentions. *Information & Management*, 44(3):253–262, 2007.

[63] J. Wu and K. Goh. Evaluating longitudinal success of open source software projects: A social network perspective. In *Proc. of the 42nd Annual Hawaii International Conference on System Sciences (HICSS)*, Waikoloa, HI, 2009.

[64] J. Wu, K.-Y. Goh, and Q. Tang. Investigating success of open source software projects: A social network perspective. In *Proc. of the International Conference on Information Systems (ICIS)*, 2007.

[65] J. Xu, S. Christley, and G. Madey. Application of social network analysis to the study of open source software. In J. Bitzer and P. J. H. Schröder, editors, *The Economics of Open Source Software Development*, chapter 11, pages 205–224. Elsevier Press, Amsterdam, The Netherlands, 2006.

[66] L. Zhao and S. Elbaum. Quality assurance under the open source development model. *Journal of Systems and Software*, 66(1):65–75, April 2003.