Socio-technical Interactions in OSS Development

Jasveen Kaur[1], Amandeep Kaur[1], Prabhjot Kaur[1]

[1]Scholar, Guru Nanak Dev University, Amritsar

E-mail-jasveenkaur.1990@gmail.com

ABSTRACT -This study is going to provide directions to open source practitioners to better organize their projects to achieve greater performance. In this research, we try to understand socio-technical interactions in a system development context by examining the joint effect of developer team structure and open source software architecture on OSS development performance. We hypothesize that developer team structure and software architecture significantly moderate each other's effect on OSS development performance. Empirical evidence supports our hypotheses and suggests that Larger teams tend to produce more favorable project performance when the project being developed has a high level of structural interdependency while projects with a low level of structural interdependency require smaller teams in order to achieve better project performance. Moreover, centralized teams tend to have a positive impact on project performance when the OSS project has a high level of structural interdependency. However, when a project has a low level of structural interdependency, centralized teams can impair project performance.

Keywords—Open source software, collaboration network, social network analysis, software architecture, software project performance, network centralization, software structural interdependency

I) INTRODUCTION

In recent years, Open Source Software(OSS) development has caused great changes in software world. The software developers collaborate voluntarily to develop software that they or their organizations need [1]. Compared with traditional software development, OSS development is unique in that it is self-organized by voluntary developers. Moreover, OSS projects automatically generate detailed and public logs of developer activities and project outputs in the form of repositories, allowing a clear view of their inner working details [1]. These unique aspects of OSS have inspired studies regarding motivations of individual participants, governance of OSS projects [3],organizational learning in OSS projects[2], architecture of OSS code [4], in OSS projects. These OSS studies have increasingly pointed toward the inseparable role of the social and the technical aspects in shaping OSS development processes and outcomes. Previous OSS studies suggest that OSS development is particularly suited for an examination of combined effects of the social and the technical in a system development context since it promotes interactions between software developers and software artifacts. This study focuses on OSS developer team structure as the social aspect and software architecture as the technical aspect of OSS projects. Our general research question is: ***what is the joint effect of developer team structure and OSS project architecture on OSS development performance?*** The answer to our research question can serve as a step towards integrating the separate lines of work on OSS development's social and technical dimensions into a coherent research literature and also helping OSS practitioners to understand the strengths and weaknesses of the OSS development process

II) INSEPARABLE ROLE OF THE SOCIAL AND THE TECHNICAL ASPECTS

Researchers have long recognized the relationship between social processes and technical properties in an organizational work context. The organizational information processing theory (OIPT) provides a widely cited perspective of this composite relationship: to achieve optimal performance, there should be a match between information processing capabilities of organizational structure (social processes) and information processing needs of a given task (technical properties)[6]. In the organizational literature, information processing capabilities are typically assessed by looking at the collaboration structure of the workforce while information processing needs are evaluated by examining the level of interdependency among task units. Cataldo et al developed a social–technical congruence measure that captures the proportion of collaboration activities actually occurred in development teams relative to the total number of collaboration activities required by interdependency among software development task assignments. A significant impact of this social–technical congruence on project productivity manifests the equally important roles of organizational structure and task characteristics in determining software project performance [9]. In addition, Kim and Umanath [7] employed a multiplicative interaction model in examining the relationship between development team structure, software task characteristics, and project performance. This modeling approach revealed that team structure and task characteristics served mutually moderating roles in affecting project performance outcomes.

In summary, both the organizational and software engineering literatures emphasize the inseparable role of organizational structure and task characteristics in organizational work performance. Taken together, prior organizational and software engineering research suggests that social processes and technical properties can play equally important and mutually moderating roles in software development performance. An understanding of the mutually moderating roles of team structures and project architecture can help OSS practitioners to realize the social and the technical aspects of OSS development altogether and harvest project performance gains from their joint effect.

III) SOCIO-TECHNICAL INTERACTIONS IN OSS

1)Social—Development Team Structure:

Open source software development is a kind of distributed software development that has a large amount of contributors and because of using Internet and make sharing freely it is so successful and useful that developers can communicate over the distance. Due to having variety of contributors in OSS projects and the story of knowledge sharing among the project can be more powerful and this might lead to improve even the position of contributors. For example they can move to the developers group from users or even in developers group move to the core developers. Core developers are a small amount of expert developers, integerated to control and manage the system. Codevelopers are the people who have directly impact on software development in the project, also they effect on the code base and they can find issues on licensing.
Following prior research, we view OSS development team structure as an important social aspect of OSS development since it manifests information processing capabilities of OSS workforce[5]. We conceptualize the development team structure according to social network theory[11]. Social network theory models individual actors as nodes of a graph joined by their relationships depicted as links between the nodes [8].When relationships are defined as collaborations on a task, the social network is specified as a collaboration network.We choose to generate collaboration networks on an intraproject level; that is, each collaboration network include developers of a single OSS project as nodes and collaboration incidences on tasks (i.e. source code files) of the same project as links.An intraproject collaboration network is a close-up view of relationship structures within a particular project. Compared with an interproject (i.e., community-level) network, an intraproject (i.e., project-level) network is more relevant to our research question since it allows us to evaluate how organizational structure of a particular OSS project team affects performance of the corresponding project.

In OSS projects, the basic unit of work is a file in the OSS distributed version control system (DVCS) like git. Hence, we view collaboration tasks as the files in the DVCS system. A collaboration incidence occurs when two developers make code commits to the same sourcecode file. A collaboration network refers to the graph made of open source developers as nodes and the collaboration incidences on the same file as links.

We characterize OSS collaboration network structure by two commonly used measures: network size and network centralization. Network size is the number of nodes in a graph. It indicates the overall scope of the collaboration network.

Network size captures the open nature of OSS development where a large number of developers collaborate to develop software in contrast to traditional software development where the number of developers is comparatively lower. Network centralization indicates the extent to which a network is centralized around one or a few nodes. Centralization of a network is measured in comparison to the most central network—the star network. In the star network, one central node connects to all of the other nodes while all other nodes are only connected to the central node. Any deviation from this structure indicates a reduction in network centralization. Although there are other metrics of collaboration network structure, we chose to focus on the two measures because network size and centralization reflect the major difference between the two alternative philosophies for organizing programming teams: the chief programmer team and the egoless team. Such teams are intentionally small and centralized around a few programming experts. In contrast, the egoless team reflects decentralized communication and collaboration among programmers and is less concerned about team size. An examination of network size and centralization of OSS teams helps us to understand the link between the unique characteristics of OSS collaboration network structure and OSS development performance.

2)Technical—Software Architecture:

In recent years, the study of software architecture (SA) has emerged as an autonomous discipline requiring its own concepts, formalisms, methods, and tools. SA represents a very promising approach since it handles the design and analysis of complex distributed systems and tackles the problem of scaling up in software engineering. Through suitable abstractions, it provides the means to make large applications manageable.

An important technical aspect of software development projects is the structural interdependency among processing elements of the software being developed. Software structural interdependency is “the strength of association established by a connection from one module to another.” [4]. In other words, software architecture is a formal way to describe the structural interdependency of a software system in terms of components and their interconnections. By decomposing the overall task into parts and then designing, implementing, or maintaining each individual part, software architecture or software structural interdependency provides a feasible way to develop and manage large systems. It reduces the complexity of software development projects to associations.

Prior research shows that software architecture is an important indicator of the information processing need of a software project. In general, the load and diversity of information needed to be processed by a software project increases with the level of structural interdependency among its processing elements.

3)Interaction Between Team Structure and Software Architecture:

In this study, interaction between the social and the technical aspects in an organizational work context refers to the mutually moderating or mutually contingent relationship between team structure (network size and centralization) and software architecture (structural interdependency) in OSS development. In other words, we conceptualize socio-technical interactions in OSS projects as multiplicative interaction of team structure and software architecture.

IV) HYPOTHESES

We develop our hypotheses following the central argument of OIPT that organizational work performance is jointly determined by information processing capabilities of the workforce and information processing needs of the task. As discussed before, structure of the team collaboration network reflects the information processing capabilities of a development team. Information processing need of the software development process is represented by the structural interdependency of the software being developed. The mutually dependent effects of development team structure and software structural interdependency on project performance are specified below :

A. Network Size and Structural Interdependency

Network size has mixed implications to information processing capabilities of a team. On one hand, larger networks incur higher coordination and communication cost.On the other hand, larger networks carry more diverse expertise and are better at specialization and division of labour among team members.The overall effect of network size on task performance depends on the structural interdependency of OSS projects. Projects with a lower level of structural interdependency do not take full advantage of the diverse expertise and perspectives in a large team while these projects have to bear the increased communication cost in such a team. Network size can therefore have a negative impact on the project performance of these projects. In a project with a high level of structural interdependency, capabilities of a large team in processing a heavy load of diverse information can produce salient project performance gains, compensating for the communication cost associated with a large team. The negative impact of network size on project performance can therefore be reduced in this scenario.

Reciprocating effects of network size on project performance, impact of software structural interdependency on project performance can vary across development teams with different network size. In traditional software development, where team size tends to be small, software structural interdependency is often found to increase software development effort which in turn can impair project performance. However, recent OSS research suggests that OSS development may resist the negative effect of software structural interdependency on development effort due to its self-organized nature. With the motive to adjust development team structure according to project characteristics, an OSS team can recruit new members when a high level of structural interdependency is perceived. This will allow the project to take advantage of information processing capabilities afforded by a large collaboration network. On the other hand, when a team is unwilling or unable to recruit additional members for a project with a high level of structural interdependency, project performance may be impaired as a result of insufficient information processing capabilities in this team. Therefore,

Hypothesis1: Collaboration network size and software structural interdependency mutually and positively moderate each other's impact on OSS project performance; that is, impact of network size on project performance is more likely to be positive when software structural interdependency is higher, and impact of software structural interdependency on project performance is more likely to be favorable when network size increases.

B. Network Centralization and Structural Interdependency

Similar to network size, network centralization has mixed effects on information processing capabilities of a team. A centralized team is better at identifying and consolidating expertise in a team. It incurs lower coordination cost than a chain-like network (low network centralization). However, a centralized team structure imposes significant information processing load on the central nodes. This can hamper the effectiveness of the whole team. Projects with a low level of structural interdependency do not require much consolidation among knowledge domains. Thus centralization of project team with low structural interdependency has no much importance, leading to suboptimal project performance. However, as the structural interdependency of a project increases, the advantage of a centralized team structure in identifying and consolidating expertise from a wider range of knowledge domains becomes important. This advantage enables a more centralized team to achieve better performance.

On the other hand, the tendency for software structural interdependency to negatively affect project performance can be particularly strong in a team with chain-like structure (low centralization) since such a team is relatively ineffective for knowledge consolidation.This tendency can be reduced by a centralized team since such a team can identify diverse information and coordinate information processing activities. Therefore,

Hypothesis 2: Collaboration network centralization and software structural interdependency mutually and positively moderate each other's impact on OSS project performance; that is, impact of network centralization on project performance is more likely to be positive in projects with a higher level of structural interdependency, and impact of software structural interdependency on project performance is more likely to be favorable when network centralization increases.

V) IMPLEMENTATION

The data for the study were collected from Github.com. Git is a distributed version control and source code management (SCM) system with an emphasis on speed. Every Git working directory is a full-fledged repository with complete history and full version tracking capabilities, not dependent on network access or a central server. When you get a copy of the repository, you do not just get the snapshot, but the whole repository itself.

A. Data Collection

From hundreds of OSS projects hosted at Github, we selected a sample of 15 projects for analysis. We selected the sample projects that were registered between Jan 2005 and Nov 2005 and that have atleast ten developers in the collaboration network. This ensures that the sampled projects have sufficient elapsed time since starting so that significant amount of development activity has already taken place. Collaboration network measures are sensitive to the size of the network. Particularly, when the network size is small, some network measures, such as centralization, become meaningless. Therefore, we have restricted our sample to projects with at least ten developers in the collaboration network.

1) *Collaboration Network Structure*: As discussed earlier, we use network size and network centralization to measure collaboration network structure. Network Size is measured by the total count of nodes in the network. The network centralization measure follows the approach proposed by Freeman [8]. It expresses the degree of inequality in a network as a percentage of that of a perfect star network of the same size. The higher the value, the more centralized the network is. We employed the widely used social network analysis software UCINET6[12] to compute the structure metrics for the collaboration networks in our sample.

2) *Software Structural Interdependency*: Although automatic tools(e.g., Lattix) are available for evaluating software architecture, these tools are usually limited to a few programming languages such as C/C++ and Java. The overwhelming amount of source code and the wide range of programming languages in our sample prevent us from measuring software structural interdependency either manually or using the automatic tool. In a study about OSS code MacCormack et al. pointed out that, "in software designs, programmers tend to group source files of a related nature into 'directories' that are organized in a nested fashion." This suggests that the Git file tree structure should be considered in gauging the level of software structural interdependency. So we measure software structural interdependency by taking the average number of source code files per folder in the Git tree. A large number of task files per folder indicates a high level of structural interdependency since files grouped into the same folder are typically related.

3) *OSS Development Performance*: In OSS projects, performance cannot be measured by parameters such as cost of development and development within schedule because OSS projects generally do not involve a budget or a deadline. Prior research on OSS development has employed various OSS project performance measures such as OSS developers' perceived project success, the percentage of resolved bugs, increase in lines of code (LoC), promotion to higher ranks in an OSS project, the number of subscribers associated with a project, and number of code commits. Among these measures, we choose the number of code commits per developer per day as our measure of OSS development performance. A code commit refers to a change in the working directory through adding, deleting, or changing software code. The number of code commits is an objective measure of project performance that has been repeatedly used by prior studies regarding effects of social processes on technical success of OSS projects. These studies indicate that the number of code commits is particularly suited for an examination of both social and technical factors in OSS development. Our measure, the number of code commits per developer per day, allows us to compare performance across projects with different team size and duration.

4) *Control Variables:* We controlled for the following variables based on the previous literature: Product Size: In the software engineering literature, product size has been identified as an important factor in manpower, time, and productivity of a project. Therefore, product size is used as a control variable. According to the literature, we measure product size as the total LoC of an OSS project.

Programming Language: Software programming language is another well-recognized factor that may affect software performance. Many projects employ more than one language. Java,C++, and C are the most frequently used. Due to the limited sample size, we created four dummy (binary) variables: "Java," "C++," and "C" to account for the top three most frequently used languages, and "other" to represent all the other languages.A project receives a value of 1 for a language dummy variable if it uses the language in consideration, a value of 0 otherwise. In other words, a project has four language values, one for each of the four dummy variables. The "other" language variable was left out of the regression model in order to prevent dummy variable trap.

License Type: OSS projects use different licensing schemes and the specific license type used may affect developer motivation and project success due to commercial or noncommercial nature of a license. License type is measured as a binary variable. All projects with the most popular OSS license, the GPL (general public license, usually indicating a noncommercial OSS product) license, are given a value of 1. All other projects have a value of 0.

B. *Data analysis*

We apply linear regression model to verify our hypotheses. Linear regression is a statistical technique for relating a dependent variable to one or more independent variables(predictors). This model employs the ordinary least squares (OLS) technique in hypothesis testing. It captures a linear relationship between the expected value of dependent variable and each independent variable (when the other independent variables are held fixed). If linearity fails to hold, even approximately, it is sometimes possible to transform either the independent or dependent variables in the regression model to improve the linearity. Here, the number of code commits per developer per day, network size and product size are log transformed to account for the nonlinear relationship between project performance and network size and product size.Other variables remain in their original form, because it is possible for them to take "0" as a value. A log transformation of these variables would arbitrarily truncate out meaningiful data points. We have used IBM SPSS statistical tool for analysis and results.

C. *Results*

The model incorporates several interaction effects. Interaction effects represent the combined effects of variables on the dependent measure. When an interaction effect is present, the impact of one variable depends on the level of the other variable. We have taken code commits per developer per day as the dependent variable and product of software structural interdependency and network size as a first interaction variable and product of software structural interdependency and network centralization as the second interaction variable. In such models, multicollinearity is a common problem.We attempted to correct for multicollinearity by centering the interacting variables on their mean. Centering just means subtracting a single value (here mean) from all of your data points. The descriptive statistics for the resulting dataset are shown in Table I. The results of the regression analysis are reported in Table II and Table III. Note that the mean values in Table I are uncentered values.

TABLE I
Descriptive Statistics

	N		Mean	Std. Deviation	Minimum	Maximum
	Valid	Missing				
codecommits	15	0	.2989	.4789	.0098	1.9000
Network size	15	0	2.6557	.8992	-1.7807	1.8893
N/w centralization	15	0	.1694	.1441	-.1495	.4305
SSI	15	0	3.9189	3.9189	-6.3811	6.0189
Product size	15	0	10.4221	1.8959	8.0380	13.2020
Licence	15	0	.4667	.5164	.0000	1.0000
C	15	0	.2000	.4140	.0000	1.0000
Cplus	15	0	.2667	.4577	.0000	1.0000
Java	15	0	.2000	.4140	.0000	1.0000

TABLE II
Model Summary

Model	R	R Square	Adjusted R Square	Std. Error of the Estimate
1	.890	.792	.272	.408635

TABLE III
RESULTS OF LINEAR REGRESSION MODEL[a]

Variable		Unstandardized Coefficients		Standardized Coefficients	Significance	Hypothesis
		B	Std. Error	Beta		
1	(Constant)	-.9857	.1309		.0433	
	SSI * Networksize	.0465	.0495	.4021	.0401	Supported
	SSI * N/w centralization	.1140	.5570	-.1419	.0848	Supported
	SSI	-.0062	.0597	-.5050	.0300	
	Network size	-.6325	.2870	1.1874	.0920	
	Network centralization	-.2036	.9232	-.6126	.0350	
	Product size	-.0579	.1115	-.2294	.0631	
	Licence	-.3221	.3766	-.3473	.0441	
	C	-.0311	.4419	-.0268	.0947	
	Cplus	.1046	.5625	.1000	.0861	
	Java	-.2838	.3973	-.2453	.0515	

a. Dependent Variable: codecommits

H1 says that collaboration network size and software structural interdependency mutually and positively moderate each other's impact on OSS project performance. This concerns the coefficient of the interaction term of network size and software structural interdependency. As shown in Table III this coefficient is positive and significant (β = 0.0465) in our model testing results. Hence H1 is supported.

Following Aiken and West [10], we calculated simple slopes of the effect of network size on the number of code commits per developer per day at three values of software structural interdependency (SSI): SSI-high = 3.918, SSI-mean = 0, SSI-low = −3.918. These three values are one standard deviation above the mean, the mean, and one standard deviation below the mean of centered SSI values, respectively.

To gain a complete view of the mutually moderating effect of network size and SSI, we also computed the simple slopes of the effect of SSI on the number of code commits per developer per day at one standard deviation above the mean (NS-high = 0.8992), the mean (NS-mean = 0), and one standard deviation below the mean (NS-low = −0.8992) values of network size.

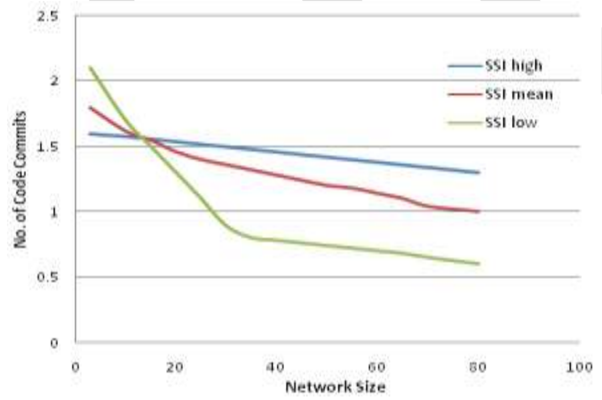

Fig. 1. Simple Slopes of Network Size

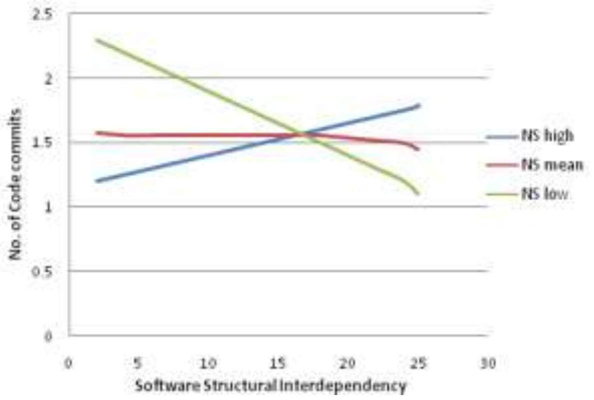

Fig.2. Simple Slopes of SSI

Fig.1 shows that network size tends to have a negative effect on project performance in terms of the number of code commits per developer per day. However, this negative effect can be mitigated by the interaction between network size and software structural interdependency since the negative simple slopes of network size become less steep as the structural interdependency values increase.

In order to find the SSI value at which the effect of network size on project performance turns from negative to positive,we equate the derivative of the number of code commits with respect to network size to zero(-0.6325+ 0.0465∗SSI =0). Since the result of this equation is centered SSI value, we add the mean of SSI to this result to gain a precise view of the inflection point. The result indicates that when there are more than 18 files per folder in a project, the effect of network size of project performance turns positive.

With respect to the effect of software structural interdependency on project performance, Fig 2 reveals that this effect is negative in small development teams but positive in large teams. Therefore, the interaction of network size and structural interdependency plays a key role in the relationship between project characteristics and project performance. By equating the derivative of the number of code commits with respect to SSI to zero (−0.0062 + 0.0465∗Network-Size = 0) and adding the mean value of network size to this result, we found that when there are more than 16 members in a project team,the effect of SSI on project performance becomes positive.

H2 proposes that collaboration network centralization and software structural interdependency mutually and positively moderate each other's impact on OSS project performance. As shown in Table III, the coefficient of the interaction term of network centralization and software structural interdependency is positive and significant (β = 0.1140). Thus, H2 is supported. Following the same simple slope finding approach for H1, we calculated the simple slopes of the effect of network centralization (NC) on the number of code commits per developer per day at SSI-high, SSI-mean, and SSI-low values .

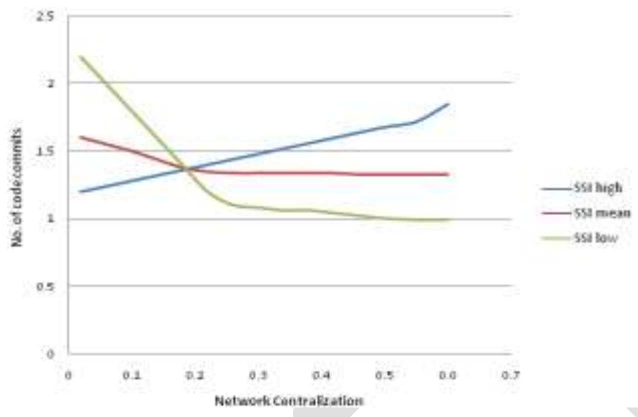

Fig. 3. Simple Slopes of Centralization.

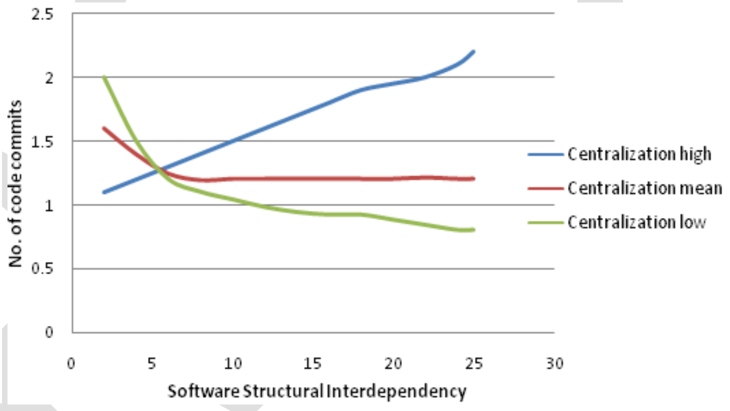

Fig. 4. Simple Slopes of SSI

Meanwhile, the simple slopes of the effect of SSI on the number of code commits per developer per day at one standard deviation above the mean (NC-high = 0.1441), the mean (NC-mean = 0), and one standard deviation below the mean (NC-low = −0.1441) values of network centralization are calculated.

Figs. 3 and 4 suggest that network centralization and software structural interdependency reciprocally remedy each other's negative effect on project performance. Setting the derivative of the number of code commits with respect to network centralization to zero (−0.2036 + 0.1140 ∗ SSI = 0), we found that at a moderate level of software structural interdependency (more than 6 CVS files per folder) the effect of network centralization turns from negative to positive. The derivative analysis of the number of code commits with respect to SSI (−0.0062 + 0.1140 ∗ Network-Centralization = 0) reveals that 0.22 degree of network centralization is the inflection point where the effect of structural interdependency turns from negative to positive.

VI) CONCLUSION

A key motivation leading us to undertake this study is the lack of research directed at socio- technical interactions in a system development context. To OSS practitioners, the main implication of our findings is that they can gain the best of both worlds by adopting a hybrid software development process that incorporates strengths of both traditional software development model and recent OSS model. Our empirical analysis demonstrates a feasible way for OSS practitioners to quantify their team structure and software architecture in order to achieve better development performance. For example, Mozilla was redesigned toward a lower level of structural interdependency so that a less focused team distributed across geographic and organizational boundaries could contribute to it. Moreover, the inflection points found in our study can be used as qualitative benchmarks for OSS practitioners to evaluate the socio-technical interactions in their projects.

REFERENCES

[1] Sajad Shirali-Shahreza,Mohammad Shirali-Shahreza, “Various Aspects of Open Source Software Development”, 2008 IEEE

[2] C. L. Huntley, “Organizational learning in open-source software projects: an analysis of debugging data,” *IEEE Trans. Eng. Manage.*, vol. 50, no. 4, pp. 485–493, Nov. 2003.

[3] E. Capra, C. Francalanci, and F. Merlo, “An empirical study on the relationship among software design quality, development effort, and governance in open source projects,” *IEEE Trans. Softw. Eng.*, vol. 34, no. 6, pp. 765–782, Nov./Dec. 2008.

[4]Caryn A. Conley, Lee Sproull, “Easier Said than Done: An Empirical Investigation of Software Design and Quality in Open Source Software Development”, Proceedings of the 42nd Hawaii International Conference on System Sciences – 2009

[5]Kevin Crowston**, “**An exploratory study of open source software development team structure”, ECIS 2003, Naples Italy

[6] J. R. Galbraith, “Organization design: An information processing view.’Interfaces, vol. 4, no. 3, pp. 28–36, 1974.

[7] K. K. Kim and N. S. Umanath, “Structure and perceived effectiveness of software development subunits: A task contingency analysis,” J. Manage.Inform. Syst., vol. 9, pp. 157–181, 1992

[8] L. C. Freeman, “Centrality in social networks: Conceptual clarification,” Social Netw., 1979.

[9] M. Cataldo, J. D. Herbsleb, and K. M. Carley, “Socio-technical congruence: A framework for assessing the impact of technical and work dependencies on software development productivity,” in Proc. Second ACM-IEEE Int. Symp. Empirical Softw. Eng. Meas., 2008, pp. 2–11.

[10] S. G.West and L. S. Aiken, Multiple Regression: Testing and Interpreting Interactions. Newbury Park, CA: Sage, 1991.

[11] G.Madey,V. Freeh, andR. Tynan, “The open source software development phenomenon: An analysis based on social network theory,” in Proc. Amer.Conf. Inform. Syst., 2002, pp. 1806–1813

[12] S. P. Borgatti,M. G. Everett, and L. C. Freeman, UCINET IV Version 1.00. Columbia, SC: Analytic Technologies, 1992