12-2019

Influence, information and team outcomes in large scale software development

Subhajit DATTA
Singapore Management University, subhajitd@smu.edu.sg

Citation

1

Published in 2019 26th Asia-Pacific Software Engineering Conference: December 2-5, Putrajaya, Malaysia: Proceedings, pp. 402-409.
DOI: 10.1109/APSEC48747.2019.00061

Influence, Information and Team Outcomes in Large Scale Software Development

Subhajit Datta
School of Information Systems (SIS)
Singapore Management University (SMU)
Singapore 178902
subhajit.datta@acm.org

***Abstract*—It is widely perceived that the egalitarian ecosystems of large scale open source software development foster effective team outcomes. In this study, we question this conventional wisdom by examining whether and how the centralization of information and influence in a software development team relate to the quality of the team's work products. Analyzing data from more than a hundred real world projects that include development activities over close to a decade, involving 2000+ developers, who collectively resolve more than two hundred thousand defects through discussions covering more than six hundred thousand comments, we arrive at statistically significant evidence indicating that concentration of information and influence in the developer communication networks of the projects are associated with the quality of a team's work products, even after controlling for various factors related to levels of developer engagement. Our results suggest that merely facilitating easy interaction between team members may not be sufficient to enhance team outcomes. The design of efficient collaborative development environments, and devising tools and processes for team assembly and governance can be informed by our results.**

***Index Terms*—Influence, interaction, software quality, team outcomes**

I. Introduction and Motivation

Raymond's *Cathedral and the Bazaar* made the provocative case for a diverse and interactive developer pool being beneficial to the outcome of large scale software development [1]. Since then, benefits of the so-called network effects are perceived to be widely prevalent in such development ecosystems [2]. Members of global development teams are encouraged to leverage collaborative development environments as they collectively design, develop, and maintain complex software systems [3], even as the benefits and challenges arising out of team sizes [4] and the distributed nature of development are being investigated [5]. Accordingly, project governance processes are oriented towards facilitating peer level interaction. Such tools and processes are meant to decentralize the flow of information and facilitate localization of influence in project teams, giving every member a level platform for shared awareness and decision-making [6]. Unfettered access to information, and equality of influence are lofty ideals. But how effective are they in practice? In this paper, we report results from a multi-project empirical study which investigates the broader context of this question. We examine 125 product teams from the Gnome suite of products[1].

Given the highly collaborative nature of open source software development, networks offer an useful way to abstract and study developer interaction [2]. A key benefit of the network paradigm is that it offers intuitive measures of several aspects of interaction; for example, whether and how one or few vertices dominate the structure and function of a particular network. In the software development context, such dominance has several implications.

While concentration of information indicates the existence of experts in the project team, concentration of influence signifies that authority is localized among few. On one hand, concentrating information and influence may be necessary to an extent for developing quality software systems within project constraints. While on the other hand, such concentration has concomitant fragility; for example, removal or impairment of the highest degree node in a highly centralized network can lead to collapse of the network's functions. We seek to examine this dichotomy through in study reported in this paper.

The **research contributions** from this study are:

- Ever since open source software development was popularized with its emphasis on interaction vis-a-vis instruction, there have been qualms on issues such as distribution of authority [7]. Such qualms can be objectively addressed in light of the empirical evidence presented in this paper on the relation between centralization of influence and information flow, and the quality of a team's work products.
- Empirical studies in software engineering are often conducted on one or few projects and the limitations of such studies are widely recognized [8], [9]. Our cohort consists of more than a hundred real world projects; hence conclusions from this study have wider relevance.
- With an increasing trend towards global collaboration, members of software teams are widely distributed across geographies and time-zones. Our results offer insights on effective governance of such teams.

In the next section, we introduce our research question, followed by an overview of related work. In subsequent sections we discuss our study setting and methodology, results with their implications and utility, and threats to validity. The paper ends with an outline of summary and conclusions from

[1] https://www.gnome.org/

this study.

II. Research Question and Hypotheses

Questions around how structural positions of individuals in a social network influence collective and individual outcomes have long interested researchers in a variety of contexts [10], [11], [12], [13], [14], [9]. In software development ecosystems, central modules have been found to be more error prone than peripheral ones [14]; informal hierarchical structures are seen to facilitate smoother coordination in distributed teams [13]; and there is evidence that developers who are more deeply embedded communication clusters of teams, perform better [9]. While these studies have focused on the individual level, we look at the role of centralization of information and influence at the level of entire networks representing intra-team developer interaction.

With this background, we investigate the research question: **How does the centralization of information and influence in a software development team relate to the quality of the team's work products?** Key terms such as "centralization", "information", and "influence" are formally defined in Section IV.

We distil the research question into the following *alternate* hypotheses, which are statistically validated in this study:

- *H1: Higher centralization of information in a software development team relates to more defects in the team's work products.* The corresponding *null* hypothesis is that there is no relation between higher centralization of information in a software development team and the number of defects in the team's work products.
- *H2: Higher centralization of influence in a software development team relates to more defects in the team's work products.* The corresponding *null* hypothesis is that there is no relation between higher centralization of influence in a software development team and the number of defects in the team's work products.

III. Related Work

We focus our discussion of related work on two areas which are most relevant to this paper: attempts at understanding the dynamics of open source software development ecosystems, and studies on the Gnome suite of products.

A. Understanding open source software development

The popularity of Linux underscored that large scale open source projects can indeed deliver quality software, within production constraints. It has been suggested that Linus Torvald's biggest contribution - even bigger than the conception of Linux itself - lies in establishing the culture of open source ecosystems that can support delivery of complex software systems [1]. Over the past decade and half, there have been many studies that seek to understand the dynamics of such ecosystems. Godfrey and Tu studied the growth, evolution, and structural change in open source software to observe that several open source systems do not seem to obey some of the Lehman's Law of software evolution [15]. Crowston et al. examined data from 7477 open source projects to understand how they function as virtual organizations, with its members focusing on competency building [16]. In a subsequent paper, Crowston and Howison investigated whether the social and communication structures of open source projects are indeed distinctive; from a study of 125 project teams, they concluded that open source project teams vary widely in their communication centralization, with some projects strongly centred on one developer to others which are significantly decentralized [12]. Muffato takes a multi-disciplinary approach towards understanding the open source phenomenon, as well as its applicability beyond software development [17]. The need to study open source software development from a multi-disciplinary perspective is further emphasized by von Krogh and Spaeth, who identify five key characteristics that distinguish this paradigm - impact, tension, transparency, communal reflexivity, and proximity [18]. Bird et al. have studied how latent social structures emerge in open source communities, using established community detection techniques; they observe that sub-communities are notably related to collaboration [8]. Merlo and Slaughter found that in open source projects, the structure of software reflects the organizational structure of the development team whereas in closed source projects, organizational structure also impacts the structure of social network of team interactions [19]. Robles, Gonzalez-Barahona, and Herraiz studied the evolution of the core set of developers in open source projects and suggested a quantitative methodology to identify how this core evolves over time [20]. In spite of the wide currency of open source development, Ancuna et al., reported that there is no globally accepted open source software development process, on the basis of a systematic study [21]. Hayashi et al. investigated whether and how developers need to collaborate in open source projects, and concluded that a committer needs to be aware of the risk of bugs being re-opened, by collaborators [22].

B. Studies on the Gnome suite of products

The development ecosystem around the Gnome suite of products has been studied from various perspectives. Koch and Schneider used data from the Gnome project to suggest an approach for effort estimation [23]. German examined the software development methods and practices used in Gnome, as well as its organizational structure in the context of its large and distributed team, and distilled a set of best practices that can be useful for other teams which try to address the challenges of global software development [24]. Lungu, Malnati, and Lanza devised a "small project observatory" to offer a visualization of the development activities of 900+ developers over 10 years of Gnome development [25]. Schackmann and Lichter evaluated the process quality in Gnome based on change request data, and presented a comparative analysis of the 25 largest products in Gnome on the basis of a quality model they develop [26]. Casebolt et al. defined the "author entropy" metric and used it to characterize author contributions per file [27]. They found evidence that larger files are more likely to have a dominant author when two authors contribute to a file. Linstead and Baldi applied latent Dirichlet allocation based techniques to mine Gnome bug reports and defined a new information-theoretic measure of coherence to estimate the quality of bug reports [28]. The presence of code clones is detected through an automated approach devised by Krinke et al., and tested on the Gnome code base [29]. The authors find that 60% of the clone pairs can be separated into original and copy. Neu et al., presented a Web-based application to support interactive visualizations for software ecosystem analysis [30]. They used their application to examine the Genome ecosystem in a bottom-up approach, and to understand how a single project and contributor can influence the entire ecosystem. Goeminne et al. presented a data-set compiling historical data about contributors to Gnome projects to complement the traditional, source code based analysis of software projects [31]. Vasilescu et al., studied the Gnome ecosystem to explore the extent to which projects and contributors specialize in particular activity types [32].

Our results complement these studies by examining the relations between centralization of information and influence, and the quality of the team's work products in large scale software development.

IV. Study Setting and Methodology

In the following subsections, we describe the context and method of our study.

A. Accessing and filtering Data

The Gnome data-set used in this study was made available for a mining challenge in a conference related to mining software repositories [33]. Gnome (also written as "GNOME") is a desktop environment composed entirely of free and open source software[2]. The data-set covers 389 Gnome "components", each of which were modules in the Gnome suite, running as related, but independent projects. In subsequent discussion, "project" will refer to the development ecosystems around each of these products, and "team" will denote the group of developers working on each project. For our analysis, we selected 125 projects which have one or more developers owning a resolved bug from the Gnome data-set. In total, these projects covered development activities over an eight-year period, involving 2313 developers, resolving 207573 defects through discussions that include 632096 comments. The projects considered in this study had the number of defects ranging from 14 to 38513, with a median of 532. We are thus able to capture a wide range of team sizes who work on these projects.

B. Generating developer communication network

As developers interact while working together to resolve bugs, they post comments on those bugs. Such co-commenting serves as an essential vehicle for disseminating awareness and directions. From these instances of co-commenting on bugs, we extract a developer communication network for each project (referred to as "network" in subsequent discussion), in the following way: The vertices (nodes) of the networks are developers; two developers are connected by an edge (undirected link) if *both* the developers have commented on at least one common bug during the active duration of the project.

C. Selecting model variables

To validate the hypotheses, we seek to use statistical models to understand how independent variables relate to the dependent variable, after accounting for the effects of the control variables. Choices of the dependent and independent variables are informed by our research question and the context of the study; control variables are selected on the basis of well recognized peripheral influences on the dependent variable. We now describe how each of the variables in our models are calculated.

1) Dependent variable: As established in literature, bug count can be taken as a proxy for the quality of a project team's work products [34]. Thus our dependent variable is ***DefectCount***, which is the number of bugs in "resolved" status for each project. In subsequent discussion, "defect" and "bug" will be used interchangeably. Figure 1 shows the boxplot of the defect counts in the projects; expectedly, the distribution of defects is right skewed, with many projects having few defects, and few project having many defects.

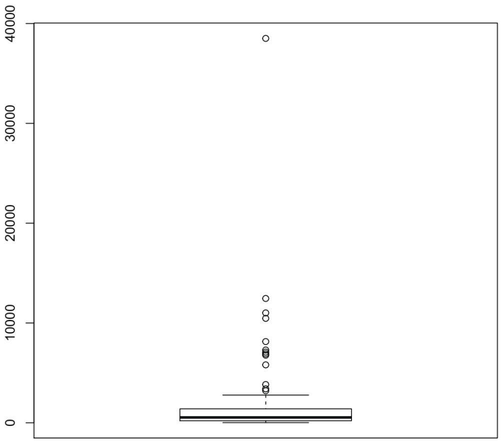

Fig. 1. Boxplot of dependent variable *DefectCount* across the projects.

2) Independent variables: These represent factors whose influence on the dependent variable is of interest to us. Degree centrality of a vertex in a network is the number of other vertices that vertex is directly connected to. Degree centrality is an indication of the *extent of information* flowing through the vertex [2]. In our network, developers with higher degrees participate in more instances of co-commenting on bugs, which facilitate their enhanced access to project information. On the other hand, eigenvector centrality indicates the *level of influence* of a vertex [2]. Calculating the eigenvector centrality

[2] https://en.wikipedia.org/wiki/GNOME

involves assigning relative scores to all nodes in the network on the assumption that connections to high-scoring nodes contribute more to the score of the node in question. Metrics such as Google's PageRank and Katz centrality are types of eigenvector centrality. For a given graph $G := (V, E)$ with $|V|$ as the number of vertices and the adjacency matrix $A = (a_{v,t})$ such that $a_{v,t} = 1$ if vertex v is linked to vertex t and $a_{v,t} = 0$ otherwise, eigenvector centrality x_v for the vertex v is given by the following definition, where λ is a constant[3] [2] :

$$x_v = \frac{1}{\lambda} \sum_{t \in G} a_{v,t} x_t$$

Intuitively, *centralization* reflects the process by which the activities of an organization become concentrated among a particular group or individual.

Defined formally, if $C_x(p_i)$ is any centrality measure of vertex i, if $C_x(p\star)$ is the largest such measure in the network, and if $max \sum_{i=1}^{N} C_x(p\star) - C_x(p_i)$ is the largest sum of differences in point centrality C_x for any graph with the same number of vertices, then the centralization[4] [2] of the network is:

$$C_x = \frac{\sum_{i=1}^{N} C_x(p\star) - C_x(p_i)}{max \sum_{i=1}^{N} C_x(p\star) - C_x(p_i)}$$

On the basis of the above definitions, we take the independent variables in our models as the degree centralization (***Information***) and the eigenvector centralization (***Influence***) of the developer co-commenting network defined earlier.

Figure 2 indicates the distribution of the degree centralization across the projects to be right skewed, with relatively few projects having high degree centralization, and many projects having relatively low degree centralization.

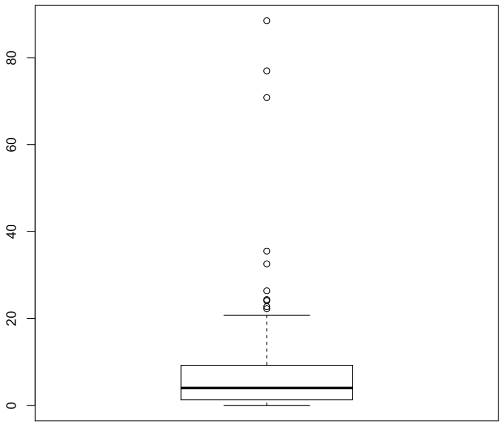

Fig. 2. Boxplot of independent variable *Information* across the projects.

From the boxplot of the eigenvector centralization in Figure 3 we observe that the distribution of this variable is left skewed, with relatively many projects having high eigenvector centralization, and few projects having relatively low eigenvector centralization.

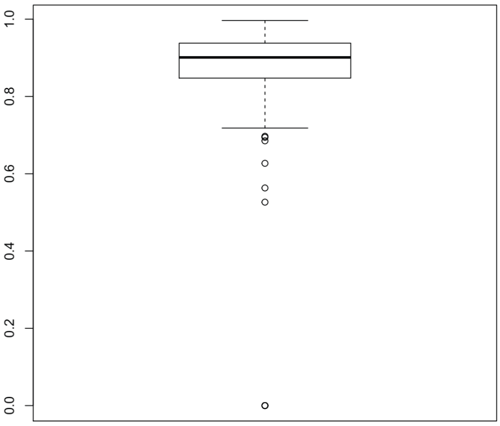

Fig. 3. Boxplot of independent variable *Influence* across the projects.

3) Control variables: These are included in the model to account for peripheral factors - other than those captured by the independent variables - that may influence the dependent variable. As the number of resolved bugs in a project is likely to be related to the number of developers owning resolved bugs, we take the latter as the ***Ownership*** control variable. Bug resolution is influenced by the extent of developer interest impinging on bugs in a project [1], thus we take the number of developers commenting on the resolved bugs as the ***Interest*** control variable. To control for the level of developer attention on individual bugs, we take the average number of comments per bug as the ***Attention*** control variable. How long developers remain engaged on resolving a bug is also expected to influence the number of bugs resolved; the average of the elapsed time in hours between the first comment by each developer on any bug and the last comment by the same developer on any bug in the project is taken as the ***Span*** control variable. We control for the extent of time developers have remained active in resolving bugs in each project as the ***Focus*** control variable, calculated as the elapsed time between first comment and last comment on bugs in a project. Other than discussions, it is important to control for the amount of development activities going around bug resolution; accordingly, the average number of activities (as defined in the data-set) around bugs in a project is taken as the ***Activity*** control variable. To control for the age of the product, we take the elapsed time between the earliest and latest creation date across all bugs in the project as the ***Age*** control variable. Bugs which are deemed more important by the developer community get more attention, to control for this effect we use the variable ***Priority***, calculated as the average of the priorities of all bugs in the project.

D. Choosing a modelling paradigm

As our dependent variable is a *count* of resolved bugs in a project, Poisson regression was first considered as a modelling

[3] http://en.wikipedia.org/wiki/Centrality#Eigenvector_centrality
[4] http://en.wikipedia.org/wiki/Centrality#Centralization

paradigm. A major threat to the validity of using Poisson regression is *overdispersion*, which is indicated by a violation of the strong underlying assumption of equality of variance and mean of a Poisson distribution [35]. On the other hand, multiple linear regression has the assumptions of linearity, normality, and homoscedasticity of the residuals, and lack of multicollinearity between the independent variables [36]. These were found to hold within reasonable limits in our study. Histogram, P-P plot, and scatter plots of the standardized residuals were used to establish the residual properties. We decided to use multiple linear regression with some of the variables suitably transformed (as explained later), as its underlying assumptions were satisfied, and it provided a high goodness of fit (see Table III).

E. Evaluating the model

TABLE I
DESCRIPTIVE STATISTICS OF THE MODEL VARIABLES

Variable name	Mean	Standard deviation
DefectCount	1660.58	3998.42
Information	0.807	0.132
Influence	0.856	0.176
Ownership	18.504	21.43
Interest	821.90	1913.79
Attention	3.37	1.05
Span	2633.22	1495.63
Focus	52920.02	15714.83
Activity	6.87	2.72
Age	58285.81	31323.74
Priority	2.16	0.188

In Table I we present the descriptive statistics of the model variables. The dependent and independent variables are transformed by taking the square root, for a better fit to the linear model. Table II shows the correlation matrix of the model variables. From this table, we observe that the correlation between the independent variables was notably low (0.156); addressing one of the underlying assumption for multiple regression.

Table III gives the details of the models. Column I specifies the *base* model showing the effects of the control variables on the dependent variable and Column II shows the *refined* model which additionally includes the independent variables. The coefficient of each control and independent variable in the regression equation is mentioned, with its standard error. The significance of each coefficient is calculated on the basis of their respective p values. The p value for each coefficient is calculated from the t-statistic (ratio of each coefficient to its standard error) and the Student's t distribution. The R^2 values give the coefficient of determination – calculated as the ratio of the regression sum of squares to the total sum of squares - indicating the goodness of fit of the regression model in terms of the proportion of variability in the data set that is accounted for by the model. We also report the degrees of freedom; the Fisher F-statistic — the ratio of the variance in the data explained by the linear model divided by the variance unexplained by the model; and the p value reflecting the overall statistical significance of the model, calculated using the F-statistic and the F-distribution. For the coefficients as well as the overall regression, if $p <$ level of significance, we conclude the corresponding result is statistically significant. The levels of significance for each of the p values are also indicated in Table III.

V. Results and Discussion

After running the regression mode we also checked the variance inflation factors of the model variables; the levels of their values indicated that multicollinearity did not present a threat to the validity of our models. Figure 4 shows the histogram of the residuals from the regression model; it is evident that the distribution is reasonably close to a normal distribution.

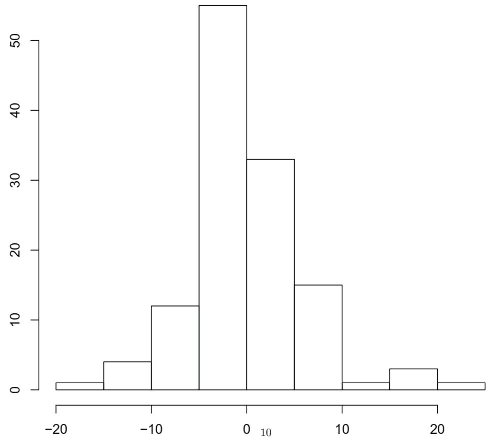

Fig. 4. The distribution of the residuals from the regression model.

With reference to Table III, we observe that both the base and refined models are statistically significant and from the R^2 values it is evident that the base model is able to explain 87.9% of the dependent variable's variability whereas the refined model explains 94.3%. Thus, inclusion of the independent variables over and above the control variables leads to an increase in the explanatory power of the model by around 7%. The increase in the F-statistic from 113 to 207 from the base to the refined model also indicates that inclusion of the independent variables lead to a better fit of the model.

Figure 5 shows the scatter plot between the actual defect count and that predicted by using the refined model. The Pearson correlation coefficient between the actual and predicted defect counts is 0.92. This strong correlation suggests the model can be used to predict the number of defects generated from teams' work products. To further establish the refined model's utility for prediction, we performed 10-fold cross validation, whose results are shown in Figure 6. Evidently, the model shows a close fit with the data, with a mean sum of squares value of 1072. Thus the refined model does not show notable effects of overfitting, and can be used effectively for predicting the defect count in other similar contexts.

TABLE II
CORRELATION MATRIX

	DefectCount	*Ownership*	*Interest*	*Attention*	*Span*	*Focus*	*Activity*	*Age*	*Priority*	*Information*	*Influence*
DefectCount	1	0.574	0.977	-0.129	-0.157	0.242	-0.118	0.457	0.368	0.259	0.11
Ownership		1	0.551	0.555	0.158	0.368	0.142	0.448	0.03	0.103	0.299
Interest			1	-0.161	-0.23	0.264	-0.16	0.543	0.395	0.208	0.108
Attention				1	0.514	0.0257	0.548	-0.055	-0.251	0.23	0.06
Span					1	0.338	0.482	0.035	-0.285	0.058	0.09
Focus						1	0.171	0.682	0.036	0.088	0.3
Activity							1	-0.047	-0.184	0.116	0.035
Age								1	0.067	0.12	0.251
Priority									1	-0.043	-0.159
Information										1	0.156
Influence											1

TABLE III
RESULTS OF REGRESSION FOR THE EFFECTS ON BUG RESOLUTION TIME.(SUPERSCRIPTS '***', '**', '*', '·' DENOTE $p \leq 0.0001$, $p \leq 0.001$, $p \leq 0.01$, $p \leq 0.05$ RESPECTIVELY)

	I *Base model*	**II** *Refined model*
Intercept	-42.7^{***} (12.2)	-35.8^{***} (9.28)
Control variables		
Ownership	0.303^{***} (0.005)	0.362^{***} (0.036)
Interest	-0.009^{***} (0.0007)	0.007^{***} (0.0005)
Attention	0.353 (1.04)	−0.864 (0.721)
Span	−0.0003 (0.0008)	0.0005 (0.0005)
Focus	0.0004 (0.0009)	0.0008 (0.0006)
Activity	0.151 (0.392)	−0.115 (0.269)
Age	$0.0009^{\cdot}$ (0.0004)	0.0008^{**} (0.0003)
Priority	24.2^{***} (5.15)	25.8^{***} (3.59)
Independent variables		
Information		4.49^{***} (0.391)
Influence		-17.4^{***} (3.69)
N	125	125
R^2	0.879	0.943
df	116	114
F	113	207
p	< 0.001	< 0.001

Focusing on the coefficients of the refined model, we see that effects of both the independent variables *Information* and *Influence* are statistically significant. From the signs of the coefficients, it is apparent that higher centralization of information relates to more defects, whereas higher centralization of influence leads to fewer defects. Thus we find empirical evidence to reject the null hypotheses corresponding to *H1* and *H2* in favour of these alternate hypotheses. The directionality of the effect as supported by the empirical evidence is as hypothesized in $H1$, but opposite to what is hypothesized in *H2*. Let us discuss the implications of this finding.

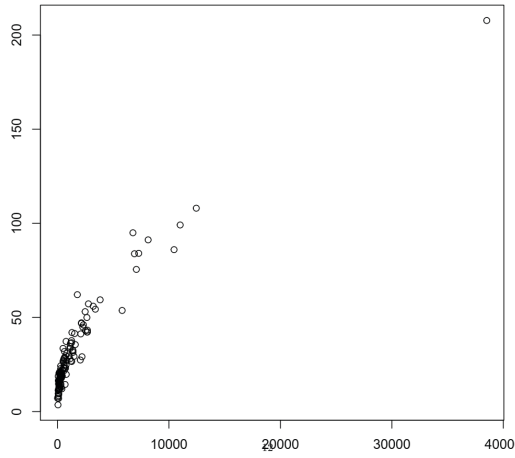

Fig. 5. Scatterplot of actual number of defects and those predicted by the regression model.

The fact that higher centralization of information relates to more defects is not unexpected. In a project, as information increasingly flows through a centralized hub, there are higher possibilities of that hub becoming a bottleneck. With information overload, parsing, processing, and disseminating information become difficult for an individual, and this may translate to information gaps elsewhere in the network. As software bugs often arise out of a lack of contextual awareness, inadequate distribution of information in a team can be seen to relate to more defects in a team's work products [34]. However, the fact that our evidence leads us to reject the null hypothesis corresponding to $H2$, and the effect of *Influence* is opposite to what is hypothesized in $H2$, is counter-intuitive. There is a general perception that, as in open societies, in open source software development too, decentralization of influence is more effective [17]. Intuitively, it also seems to make more sense that rather than one highly influential member holding sway over the entire team, it will be better for the team to have each member having adequate influence on her immediate sphere of interest. How then do we explain our result that higher centralization of influence relates to fewer defects?

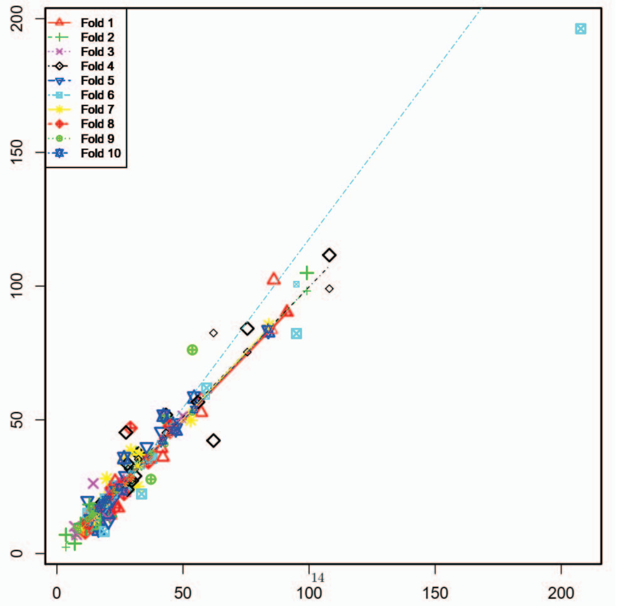

Fig. 6. Results from 10-fold cross validation.

The contrasting directionality of the associations between the independent variables and the dependent variable points to an interesting dichotomy in software development involving large and diverse teams. As tools and processes increasingly support easy interaction between team members, there is a possibility that developers become overwhelmed by the peripheral noise such interaction essentially entails. Often the development ecosystem in such situations can become a "play-pen for developers" with their collective effort *not* converging to desired project outcomes [37]. Such possibilities are especially strong when the team relies notably on remote collaboration, as is the case in our study setting [38]. In these situations, a strong central influence to set the project's directions, track and steer progress, and ensure quality constraints are met, is necessary. Lack of such a centralized influence can relate to a degradation in the quality of the work product, whose evidence we see in our study. Recent results from the burgeoning studies of "team science" are congruent with this finding [39].

Our results can inform the design of collaborative software development tools and processes in various ways. This study indicates that it is important to ensure each developer is able to access relevant project information directly. But concomitantly, it also points to the need to have a governance mechanism where influence is clearly defined and concentrated rather than being loosely diffused in the team. This has several implication in the assembly and governance of distributed development teams. With the known overheads of communication in software projects it is thus important that teams have resident experts in each location, so that developers will not need to reach out far to gain vital project awareness [40]. Also, influential members of the team need to make a special effort to reach out other members, so that authority gradient does not pose a barrier to intra-team communication. Our results highlight the need to balance concentration of influence with diffusion of authority in a team. This can be achieved with planned sensitization and training of team members.

VI. Threats to Validity and Future Work

Threats to **construct validity** are concerned with measurement of variables. As mentioned earlier, our independent variables are calculated as network measures and the control variables are derived from the Gnome data-set. While the models may be augmented in different ways, errors are unlikely to be present in the measurements of the variables currently included in our model. **Internal validity** ensures a study is free from systematic errors and biases. As the Gnome data-set is our only source of data, our study is relatively free from this threat. Whether the results from a study are generalizable is associated with the threat of **external validity**. Our results are based on studying many projects from the same Gnome platform. Thus we do not claim our results to be generalizable without further investigation. **Reliability** is concerned with the reproducibility of results. Given access to the data, our results are reproducible. In addition to the above threats, we recognize that this is an observational study rather than a controlled experiment; thus in the statistical models presented, correlation does not imply causation. In our future work we plan to repeat this study across other data-sets so that we are able to generalize our finding. Additionally, we plan to complement our quantitative approach with a qualitative one based on interviews and surveys, which will give us further insights on the observed effects.

VII. Summary and Conclusions

In this paper we presented results from an empirical study of 125 projects from the Gnome suite. We examined how centralization of information and influence in project teams relate to the number of defects in the team's work products. We found statistically significant evidence that higher centralization of information relates to more defects, whereas higher centralization of influence is associated with fewer defects. Within the scope of this study, we conclude that decentralization of information flow in a team is beneficial to a team's output. However, concentration of influence in the team - contrary to a widely held perception - helps rather than hinders the production of quality work products. Our results can inform project governance and quality assurance initiatives in large scale software development.

VIII. Acknowledgement

The author thanks Mr N. Mulianto and Mr S. Acharya for their help with processing some parts of the data-set used in this study.

References

[1] E. S. Raymond, *The Cathedral and the Bazaar: Musings on Linux and Open Source by an Accidental Revolutionary*. O'Reilly, 2001.

[2] M. O. Jackson, *Social and Economic Networks*. Princeton, NJ: Princeton University Press, Nov. 2010.

[3] V. Gilrane, "Working together when we are not together," https://www.blog.google/inside-google/working-google/working-together-when-were-not-together/ Last accessed: April 11, 2019, 2019.

[4] L. Wu, D. Wang, and J. A. Evans, "Large teams develop and small teams disrupt science and technology," *Nature*, vol. 566, no. 7744, p. 378, Feb. 2019. [Online]. Available: https://www.nature.com/articles/s41586-019-0941-9

[5] S. Datta, "How does developer interaction relate to software quality? an examination of product development data," *Empirical Software Engineering*, vol. 23, no. 3, pp. 1153–1187, Jun. 2018. [Online]. Available: https://doi.org/10.1007/s10664-017-9534-0

[6] S. Baltes and S. Diehl, "Towards a theory of software development expertise," in *Proceedings of the 2018 26th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*, ser. ESEC/FSE 2018. New York, NY, USA: ACM, 2018, pp. 187–200. [Online]. Available: http://doi.acm.org/10.1145/3236024.3236061

[7] K. Beck and B. Boehm, "Agility through discipline: A debate," *Computer*, vol. 36, no. 6, pp. 44–46, 2003.

[8] C. Bird, D. Pattison, R. D'Souza, V. Filkov, and P. Devanbu, "Latent social structure in open source projects," in *Proceedings of the 16th ACM SIGSOFT International Symposium on Foundations of software engineering*, ser. SIGSOFT '08/FSE-16. New York, NY, USA: ACM, 2008, pp. 24–35.

[9] K. Ehrlich and M. Cataldo, "All-for-one and one-for-all?: a multi-level analysis of communication patterns and individual performance in geographically distributed software development," in *Proceedings of the ACM 2012 conference on Computer Supported Cooperative Work*, ser. CSCW '12. New York, NY, USA: ACM, 2012, pp. 945–954.

[10] R. T. Sparrowe, R. C. Liden, S. J. Wayne, and M. L. Kraimer, "Social networks and the performance of individuals and groups," *The Academy of Management Journal*, vol. 44, no. 2, pp. 316–325, Apr. 2001, ArticleType: research-article / Full publication date: Apr., 2001 / Copyright © 2001 Academy of Management.

[11] K. Börner, L. Dall'Asta, W. Ke, and A. Vespignani, "Studying the emerging global brain: Analyzing and visualizing the impact of co-authorship teams: Research articles," *Complexity*, vol. 10, p. 57–67, Mar. 2005, ACM ID: 1067136.

[12] K. Crowston and J. Howison, "The social structure of free and open source software development," *First Monday, ISSN 1396-0466*, Feb. 2005, the authors examine communication patters of FLOSS projects, finding that FLOSS development teams vary widely in centralizing or decentralizing their communications.

[13] P. Hinds and C. McGrath, "Structures that work: social structure, work structure and coordination ease in geographically distributed teams," in *Proceedings of the 2006 20th anniversary conference on Computer supported cooperative work*. Banff, Alberta, Canada: ACM, 2006, pp. 343–352.

[14] M. Pinzger, N. Nagappan, and B. Murphy, "Can developer-module networks predict failures?" in *Proceedings of the 16th ACM SIGSOFT International Symposium on Foundations of software engineering*. Atlanta, Georgia: ACM, 2008, pp. 2–12.

[15] M. Godfrey and Q. Tu, "Growth, evolution, and structural change in open source software," in *IWPSE '01: Proceedings of the 4th International Workshop on Principles of Software Evolution*. New York, NY, USA: ACM Press, 2001, pp. 103–106.

[16] K. Crowston and B. Scozzi, "Open source software projects as virtual organisations: competency rallying for software development," *Software, IEE Proceedings -*, vol. 149, no. 1, pp. 3–17, 2002.

[17] M. Muffatto, *Open Source: A Multidisciplinary Approach*. Imperial College Press, 2006.

[18] G. von Krogh and S. Spaeth, "The open source software phenomenon: Characteristics that promote research," *J. Strateg. Inf. Syst.*, vol. 16, no. 3, pp. 236–253, 2007.

[19] F. Merlo and S. A. Slaughter, "The co-evolution of organizational structures and software structures in open source and closed source projects," Technology and Operations Management Seminars 2009-2010, Harvard Business School, 2009.

[20] G. Robles, J. Gonzalez-Barahona, and I. Herraiz, "Evolution of the core team of developers in libre software projects," in *Mining Software Repositories, 2009. MSR '09. 6th IEEE International Working Conference on*, 2009, pp. 167–170.

[21] S. Acuna, J. Castro, O. Dieste, and N. Juristo, "A systematic mapping study on the open source software development process," in *Evaluation Assessment in Software Engineering (EASE 2012), 16th International Conference on*, May 2012, pp. 42–46.

[22] H. Hayashi, A. Ihara, A. Monden, and K.-i. Matsumoto, "Why is collaboration needed in oss projects? a case study of eclipse project," in *Proceedings of the 2013 International Workshop on Social Software Engineering*, ser. SSE 2013. New York, NY, USA: ACM, 2013, pp. 17–20.

[23] S. Koch and G. Schneider, "Effort, co-operation and co-ordination in an open source software project: GNOME," *Information Systems Journal*, vol. 12, no. 1, pp. 27–42, Jan. 2002.

[24] D. M. German, "The GNOME project: a case study of open source, global software development," *Software Process: Improvement and Practice*, vol. 8, no. 4, pp. 201–215, Oct. 2003.

[25] M. Lungu, J. Malnati, and M. Lanza, "Visualizing Gnome with the Small Project Observatory," in *6th IEEE International Working Conference on Mining Software Repositories, 2009. MSR '09*, May 2009, pp. 103–106.

[26] H. Schackmann and H. Lichter, "Evaluating process quality in GNOME based on change request data," in *6th IEEE International Working Conference on Mining Software Repositories, 2009. MSR '09*, May 2009, pp. 95–98.

[27] J. Casebolt, J. L. Krein, A. MacLean, C. D. Knutson, and D. Delorey, "Author entropy vs. file size in the gnome suite of applications," in *6th IEEE International Working Conference on Mining Software Repositories, 2009. MSR '09*, May 2009, pp. 91–94.

[28] E. Linstead and P. Baldi, "Mining the coherence of GNOME bug reports with statistical topic models," in *6th IEEE International Working Conference on Mining Software Repositories, 2009. MSR '09*, May 2009, pp. 99–102.

[29] J. Krinke, N. Gold, Y. Jia, and D. Binkley, "Cloning and copying between GNOME projects," in *2010 7th IEEE Working Conference on Mining Software Repositories (MSR)*, May 2010, pp. 98–101.

[30] S. Neu, M. Lanza, L. Hattori, and M. D'Ambros, "Telling stories about GNOME with Complicity," in *2011 6th IEEE International Workshop on Visualizing Software for Understanding and Analysis (VISSOFT)*, Sep. 2011, pp. 1–8.

[31] M. Goeminne, M. Claes, and T. Mens, "A historical dataset for the Gnome ecosystem," in *2013 10th IEEE Working Conference on Mining Software Repositories (MSR)*, May 2013, pp. 225–228.

[32] B. Vasilescu, A. Serebrenik, M. Goeminne, and T. Mens, "On the variation and specialisation of workload—A case study of the Gnome ecosystem community," *Empirical Software Engineering*, vol. 19, no. 4, pp. 955–1008, Feb. 2013.

[33] A. Hindle, I. Herraiz, E. Shihab, and Z. M. Jiang, "Mining Challenge 2010: FreeBSD, GNOME Desktop and Debian/Ubuntu," in *2010 7th IEEE Working Conference on Mining Software Repositories (MSR)*, May 2010, pp. 82–85.

[34] A. G. Koru and H. Liu, "Building defect prediction models in practice," *IEEE Softw.*, vol. 22, no. 6, p. 23–29, Nov. 2005.

[35] D. Barron, "The analysis of count data: Overdispersion and autocorrelation." *Sociological methodology*, vol. 22, pp. 179–220, 1992.

[36] B. Tabachnick and L. Fidell, *Using Multivariate Statistics*. Boston: Pearson Education, 2007.

[37] G. Booch, J. Rumbaugh, and I. Jacobson, *The Unified Modeling Language User Guide, Second Edition*. Addison-Wesley, 2005.

[38] F. P. Brooks, "The design of design," http://weatherhead.case.edu/requirements/Brooks_Plenary_0604.ppt, 2000.

[39] A.-L. Barabasi and R. V. Anderson, *The Formula: The Universal Laws of Success*, unabridged edition ed. Blackstone Pub, Nov. 2018.

[40] F. P. Brooks, *The Mythical Man-Month: Essays on Software Engineering, 20th Anniversary Edition*. Addison-Wesley, 1995.