MIS Quarterly

RESEARCH ARTICLE

FOLLOWERSHIP IN AN OPEN-SOURCE SOFTWARE PROJECT AND ITS SIGNIFICANCE IN CODE REUSE

Qiqi Jiang
Department of Digitalization, Copenhagen Business School, Howitzvej 60,
Frederiksberg 2000 DENMARK {qj.digi@cbs.dk}

Chuan-Hoo Tan
Department of Information Systems and Analytics, National University of Singapore,
13 Computing Drive, 117417 SINGAPORE {tancho@comp.nus.edu.sg}

Choon Ling Sia
Department of Information Systems, City University of Hong Kong, Tat Chee Avenue 83, Hong Kong, SAR CHINA {iscl@cityu.edu.hk}
and Department of Information Management, National Taiwan University, Taipei City 106, Taiwan R.O.C. {iscl@cityu.edu.hk}

Kwok-Kee Wei
Department of Information Systems and Analytics, National University of Singapore,
13 Computing Drive, 117417 SINGAPORE {weikk@comp.nus.edu.sg}

Appendix

Summary of Related Studies Examining OSS Leadership

Table A1. Seminal Studies Examining Leadership in the OSS Projects or Communities

Study	Methods (and Data)	Key Findings and Argument in Leadership
Fielding (1999)	Case study: Case study of Apache project.	Different from an authoritative leadership in most OSS projects, the author unveiled a shared leadership mechanism in Apache project in which a small group of controlling members uses a relatively democratic decision process.
Lerner and Tiróle (2002)	Conceptual and literature reviews.	The performance and success of an OSS project are dependent on the presence of a credible leader or leadership. They suggested that a strong centralization of authority and leadership characterizes OSS projects, such as Linux.
Fleming and Waguespack (2007)	Quantitative investigation: Archival data from Internet Engineering Task Force documents.	This study identified and discussed two imperative capabilities of OSS leadership: strong technical contribution and binding multiple communities.

Table A1. Seminal Studies Examining Leadership in the OSS Projects or Communities

Study	Methods (and Data)	Key Findings and Argument in Leadership
O'Mahony and Ferraro (2007)	Mixed method. Qualitative investigation: Interview with Debian contributors. Quantitative investigation: Regression on the variables constructed from multisourced archival data including project directory, developer database, bug-tracking database, package popularity database, and identity authentication database.	This work discussed the role of leadership in governing an OSS project. The authority of leadership can be highlighted in the bureaucratic mechanism, which increases the management efficiency. The leaders can construct the democratic governance mechanism to represent the community's interests and adapt with members' evolving interpretation of leadership.
Giuri et al. (2008)	Quantitative investigation: Econometric analysis with three-year archival data from Sourceforge.net.	OSS project leader with diversified skill set can coordinate the contribution from various participants and motivate such participants. The presence of leadership is associated with the degree of modularity of the development process.
Li et al. (2012)	Quantitative investigation: Structural equation modeling with survey data contributed by 187 developers from Sourceforge.net.	Leader can use transformational leadership and active management style to positively stimulate developers' intrinsic and extrinsic motivations, respectively.
Faraj et al. (2015)*	Quantitative investigation: Archival data collected from three different Usenet newsgroups.	This work argued that the leadership in community-based production groups can be identified by (1) extended contributed knowledge; (2) sociability, that is, facilitating communication among online participants; and (3) structural social capital, that is, positions in the online social network.
Johnson et al. (2015)	Quantitative investigation: Using survey to identify the leader of each community; applying Natural Language Processing package to extract primary variables from archival data.	This work argued that the following factors are instrumental in forming leadership: (1) occupying a formally authoritative role in community-based production groups, (2) being better positioned in the communication network, and (3) using unique patterns of language.

*Although Usernet newsgroups are not typical OSS project groups, the nature of these groups in this study, namely, object-oriented programming, databases, and C++ programming, are similar to those of OSS. The authors also cited OSS studies when they developed their theoretical arguments. Thus, we included this work in this table.

Post Hoc Analyses

In the first set of *post hoc* analyses, the data were subjected to two conventional robustness checks, namely, varying the operationalization of the dependent variable and varying the regression analysis models. We varied the dependent-variable operationalization from $AvgDiff\_CodeReuse_i$ (i.e., the average difference in code reuse among leader i's OSS projects at T_2 and T_1) to $AvgRatio\_CodeReuse_i$. This variable signifies the average change (as a ratio) in the extent of code reuse for leader i's projects at T_2 compared with that at T_1. A high value of $AvgRatio\_CodeReuse_i$ indicates a great extent of code reuse among that leader's OSS projects. Multilevel regression was applied to estimate the coefficients. Two additional regression models were used to estimate the relationship between the predictors and the focal dependent variables, $AvgDiff\_CodeReuse_i$. Specifically, we applied this variable to reconcile the effect of nuisance parameters (Model 2 in Table A2) and estimate the influence of key predictors and the related control variables for $AvgDiff\_CodeReuse_i$ with random-intercept model (Model 3 in Table A2) (McCulloch and Neuhaus 2001). Table A2 presents the summary of results. The results show significant consistency with the earlier findings obtained in the main testing.

Table A2. First *Post Hoc* Analysis

Dependent Variables	$AvgRatio\_CodeReuse_i$	$AvgDiff\_CodeReuse_i$	
	Model 1	**Model 2**	**Model 3**
$Developer\_followership_{ijt1}$	−0.088** (0.039)	−0.197** (0.085)	−0.200** (0.085)
$Observer\_followership_{it1}$	0.061*** (0.008)	0.221*** (0.018)	0.211*** (0.018)
$Project\_age_{jit1}$	−0.017 (0.031)	0.076 (0.069)	0.053 (0.071)
$Leader\_age_{ijt1}$	−0.756*** (0.104)	−1.515*** (0.253)	−1.544*** (0.241)
$Leader\_following_{ijt1}$	0.004 (0.009)	−0.062*** (0.018)	−0.062*** (0.018)
$Project\_size_{jit1}$	0.002 (0.009)	−0.012 (0.019)	−0.021 (0.020)
$Project\_knowledge_{jit1}$	−0.128*** (0.039)	−0.243*** (0.090)	−0.249*** (0.088)
$Cumulative\_Version_{jit1}$	−0.012 (0.008)	−0.024 (0.019)	−0.010 (0.019)
$Ratio_{jit1}$	−0.010 (0.014)	0.096** (0.040)	0.103*** (0.036)
$Recency_{jit1}$	−0.036 (0.012)	−0.035 (0.025)	−0.025 (0.025)
$Project\_observer_{jit1}$	0.086*** (0.011)	0.144*** (0.026)	0.183*** (0.027)
$Language\_Popularity_{jit1}$	−0.416 (0.430)	0.325 (0.905)	−0.174 (0.946)
$Language\_Media\_Coverage_{jit1}$	−0.006 (0.005)	0.005 (0.011)	0.005 (0.012)
Intercept	5.774*** (0.859)	11.388*** (2.053)	11.613*** (1.951)
Deviance (-2 × log-likelihood)	1000.527	2398.375	2356.345
Akaike's information criterion	1082.527	2484.375	2438.345
Bayesian information criterion	1284.557	2693.654	2637.889

*p ≤ 0.1; **p ≤ 0.05; ***p ≤ 0.01; figures are displayed in the format of "Coefficient (Std. Err.)"

Note: For Model 1, we replaced the dependent variable by $AvgRatio\_CodeReuse_i$ (average change as a ratio in the extent of code reuse for leader i's projects at T2 compared with T1). For Model 2, we applied restricted maximum likelihood estimation for reconciling the effect of nuisance parameters. For Model 3, we applied the random-intercept model for estimating the coefficients. The results from the three models are consistent with those in Table 4 in the main body. To easily interpret the results, we scaled down $Project\_age_{jit1}$, $Ratio_{jit1}$, and $Recency_{jit1}$ by factors of 1000, 100, and 100, respectively. The programming languages, denoted by $Programming\_Language_{ji}$, and project category, denoted by $Project\_Category_{ji}$, were created as dummy variables and estimated in the regression model but are not reported for brevity.

In the second set of *post hoc* analyses, we segmented the OSS projects into affiliated categories that could have highly diverse complexity in OSS development (Au et al. 2009; Sen 2007). The dataset contained three project categories: Software Framework, Application Software, and Documentation File. Two interesting findings can be drawn from the results in Table A3. First, a large number of followers play the role of observer; specifically, a close observer of the leader can enhance code reuse across the types of OSS projects. Second, given that the Software Framework (Model 1 in Table A3) and Application Software (Model 2 in Table A3) types are hosted on GitHub, developers have to constantly contribute. Therefore, the developer–follower relationship has a significant positive effect on code reuse. Miscellaneous OSS textual files, such as Cascading Style Sheet files for websites or unified configuration files for software, were included in Model 3 in Table A3. In contrast to the previous two categories (whose operations heavily rely on developers), the OSS projects in Documentation File can be used extensively. However, developers in this study show minimal interest in such relatively insignificant projects. Consequently, the results in this third case show that $Developer\_followership_{ijt1}$ does not influence the extent of code reuse.

Table A3. Second *Post Hoc* Analysis

Dependent Variables	$AvgDiff\_CodeReuse_i$		
	Model 1	**Model 2**	**Model 3**
$Developer\_followership_{ijt1}$	−0.193* (0.119)	−0.357** (0.147)	0.121 (0.172)
$Observer\_followership_{it1}$	0.244*** (0.026)	0.249*** (0.031)	0.210*** (0.035)
$Project\_age_{jit1}$	0.050 (0.103)	−0.002 (0.121)	−0.011 (0.128)
$Leader\_age_{ijt1}$	−4.569*** (0.425)	−0.892* (0.529)	−0.080 (0.335)
$Leader\_following_{ijt1}$	−0.069*** (0.025)	−0.046 (0.032)	−0.058 (0.036)
$Project\_size_{jit1}$	−0.030 (0.026)	−0.006 (0.032)	−0.008 (0.051)
$Project\_knowledge_{jit1}$	−0.147 (0.112)	−0.650*** (0.183)	−0.179 (0.175)
$Cumulative\_Version_{jit1}$	−0.010 (0.026)	−0.063* (0.034)	0.044 (0.038)
$Ratio_{jit1}$	0.014 (0.054)	0.231* (0.119)	0.196*** (0.050)
$Recency_{jit1}$	−0.025 (0.035)	0.006 (0.042)	−0.046 (0.056)
$Project\_observer_{jit1}$	0.197*** (0.037)	0.170*** (0.052)	0.078 (0.052)
$Language\_Popularity_{jit1}$	0.209 (1.352)	−1.936 (1.648)	−0.790 (1.802)
$Language\_Media\_Coverage_{jit1}$	−0.004 (0.015)	0.035 (0.021)	0.012 (0.024)
Intercept	34.075*** (3.196)	6.652* (3.937)	0.732 (2.475)
Deviance (−2 × log-likelihood)	1123.358	545.363	541.493
Akaike's information criterion	1193.358	609.5363	605.149
Bayesian information criterion	1339.441	722.6047	714.748

*p ≤ 0.1; **p ≤ 0.05; ***p ≤ 0.01; figures are displayed in the format of "Coefficient (Std. Err.)"

Note: We separated the samples into three sub-samples by the software categories, namely, Software Framework (Model 1), Application Software (Model 2), and Documentation Files (Model 3). To easily interpret the results, we scaled down $Project\_age_{jit1}$, $Ratio_{jit1}$, and $Recency_{jit1}$ by factors of 1000, 100, and 100, respectively. The programming languages, denoted by $Programming\_Language_{ji}$, were created as dummy variables and estimated in the regression model but are not reported for brevity.

The third set of *post hoc* analyses was conducted to check for potential multicollinearity problems. In particular, the analysis used the ordinary least squares model. In this step, we also calculated VIF and condition index (CI) values to assess potential multicollinearity problems. The multicollinearity in the stated results should not be a problem because no VIF calculations exceed 5.0, and the CI value is 24.838 (Ho and Richardson 2013). The robustness of the model was also validated and demonstrated by adding the control variables into the main model by using a stepwise approach. Due to the page limitation, the details of estimated results are available from the authors upon request.

The fourth set of *post hoc* analyses was conducted to rule out an alternative explanation for "follower inbreeding" among certain OSS projects. The results discussed in the preceding paragraphs indicate that the followership conceived through prior collaboration is not conducive to code reuse. However, these findings may be the result of an alternative explanation: enhanced code reuse may not be attributed to the open developer collaboration but can be due to the scarce participation of certain developers in other OSS projects. Their lack of enthusiasm may be caused by extreme loyalty to specific leaders. Thus, the extant projects with tight developer–follower relationships were split into two

subgroups, namely, those who worked only on a leader's project and those who worked on multiple projects with different leaders. If no difference exists in code reuse between the two groups, then the concern of "follower inbreeding" can be minimized.

Four steps were taken to test this possibility at the project level. First, a project-level dependent variable, $Diff\_CodeReuse_j$, was constructed. This variable denotes the difference in the extent of code reuse of the OSS project j managed by the same project leader between T_1 and T_2. Next, a new binary variable, $Single\_leader_{jtl}$, was constructed. This variable indicates whether the focal OSS project j is administered by a single leader (a value of 1 indicates that the project was administered by a single leader). Third, using the descriptive statistics of $Developer\_followership_{ijtl}$ (mean = 0.320 and S.D. = 0.371) presented in Table 2, the OSS projects with an extent of collaborative followership greater than 0.3 (a similar value to the sum of the mean and S.D.) were labeled as projects with a tight developer–follower relationship. A high value of $Developer\_followership_{ijtl}$ indicates a tight relationship. Finally, the control variables were transformed into the project level[1] and entered as $Single\_leader_{itl}$ into the OLS regression model. Although the selected OSS projects can be statistically accepted as the OSS projects with high developer followers ($Developer\_followership_{ijtl}$ is greater than 0.3), the regression models were consistently run with different samples through a repeated increase in the value of $Developer\_followership_{ijtl}$ from 0.3 to 0.9 by intervals of 0.1. The estimated coefficient of $Single\_leader_{jtl}$ is insignificant under different extent of collaborative followership. Thus, we can conclude that no difference exists in code reuse between the two groups. Due to the page limitation, the details of estimated results are available from the authors upon request.

Additional Investigations: Survey and Focus Groups

We conducted two *post hoc* investigations in collaboration with an IT research agency to validate our arguments regarding code reuse and the underlying role of learning. The first *post hoc* investigation included a survey, which had 164 complete responses from GitHub-registered members. The second *post hoc* investigation included focus groups of 24 GitHub-registered members.[2] Three focus group were held with eight GitHub-registered members in each (for approximately 90–120 minutes each).[3] The qualitative data[4] extracted from the focus groups enriched the survey findings. Below we present results from both investigations.

We grounded the survey in the social learning framework (Bandura 1977). The aim was to test our conjecture that code reuse (as measured in the survey as the intention to reuse the codes) can be contributed by followers who are learning from their leaders. Figure A1 presents the model. Intention to code reuse is the outcome variable. We included several control variables related to OSS values and beliefs, namely, the perception of importance to (1) code reuse, (2) sharing mentality, and (3) OSS team diversity, tenure in GitHub, OSS usage experience, gender, and age. Table A5 shows the operalization of each construct. We used SmartPLS 2.0.M3 to test the model. Table A4 reports the estimated path coefficients, and the results for construct validity and reliability.

[1] $Leader\_age_{ijtl}$ (age of leader i) and $Leader\_following_{ijtl}$ (the number of people followed by leader i) were excluded because the two variables were typically leader-level covariates.

[2] All interviewees first introduced themselves and shared their OSS experience. They were then asked about their general understanding of OSS beliefs and values: two key components of OSS ideology, code reuse in software development (especially for the OSS project), and their experience as the followers (as developers or observers) in one or more OSS projects. The group interviews were recorded in an audio–visual format. Each interviewee received a compensation of US$50 for his or her time.

[3] A total of 24 interviewees were involved, including active OSS participants with numerous experiences in OSS development and relatively passive OSS participants who mostly lurk in the community. These 24 interviewees (14 males and 10 females) were arranged into three focus groups to hold further discussions. The average programming experience (in the real project) of these respondents is approximately 5 years. Their programming language ranges from objective-oriented languages (e.g., Java, Objective-C, or C#) to scripting language (e.g., PHP, JavaScript, Python, or Ruby).

[4] Two independent assistants were recruited to transcribe the protocols of the focus groups. Two coders were then recruited to code the protocols following standard procedures (Someren et al. 1994). Their coding was very similar (Cohen's kappa = 0.88, $p < 0.010$), indicating high inter-coder reliability (Cohen and Reed 2006). Respondents were first asked to describe their understanding of OSS, general thoughts on code reuse, and personal opinions about the follower–leader relationship. Approximately 12,000 common words in the answers of each focus group discussion were found. Table A6 summarizes some representative codes and descriptions.

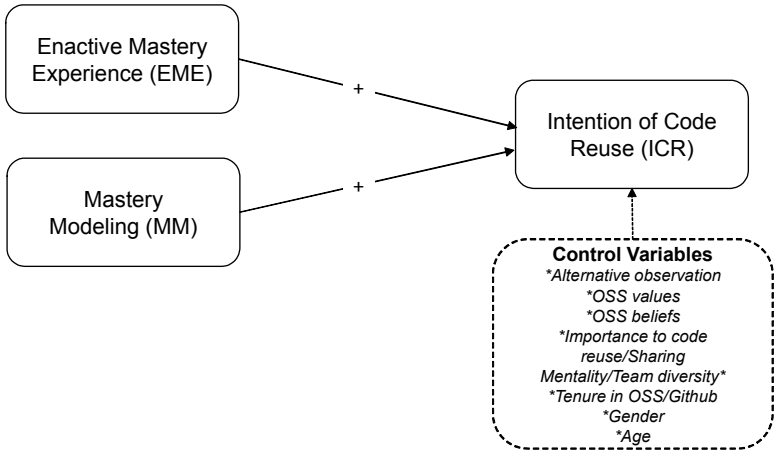

Figure A1. Conceptual Model of the Follow-up Survey

Table A4. Results of the *Post Hoc* Survey Investigation

	Estimated Path Coefficient	AVE	Composite Reliability	Cronbach's Alpha
MM	0.15**	0.638	0.876	0.815
EME	0.24***	0.609	0.817	0.701

R square = 0.59; *p value ≤ 0.1; **p value ≤ 0.05; ***p value ≤ 0.01

Table A4 shows that mastery modeling (MM) and enactive mastery experience (EME) significantly relate to the intention to code reuse. This finding provides empirical support for our suggestion that learning from OSS followers can explain the relationship between followership and code reuse. Specifically, developers as followers can obtain an in-depth understanding of the OSS projects. This situation renders them competent in applying the source codes or modular code components from their endeavoring projects in their other OSS projects. As one interviewee (FG-3-E[5]) stated, "*By working closely with the (OSS project) leader, I can have a clear image of the modular architecture of that focal project. If I need, I can directly reuse the modules or source codes (from certain modules) for other OSS projects that I work on. This really saves me a lot of effort; namely I can avoid reinventing the wheel.*"

In addition, regarding EMEs, two interesting findings were also unveiled. First, EME is developed through interaction with the leaders. As one interviewee (FG-3-H) stated, "*I received the response from the leader regarding my editing in the source codes. The approval (of my editing) conveyed the message of endorsement (of my work). Even my changes were rejected; I mostly received the comments (posted by leaders) indicating the reasons for rejection. Such interactive activities really improved my coding capability. This feedback is a personal instructor.*" Second, EME can be cultivated through collaborative diversity, as noted by another interviewee (FG-2-C): "*I can learn more through working with different people. There are breakthrough results from diversity. I don't think it is a good idea to constantly subordinate myself to the same people like in my daily employment.*"

The observers as followers are updated on the activities of their interested (OSS project) leaders, which helps them acquire miscellaneous information about software development and project management. Consequently, these observers will apply this information to the other projects. For example, an interviewee (FG-2-G) indicated, "*I follow some skilled leaders of OSS projects in GitHub to learn from them. This provides me [with] great opportunities to learn the codes they wrote and the way they managed the projects. Sometimes, I seek the codes I need from these people's activities.*" Such an opinion is also echoed by another interviewee (FG-1-A): "*Reputable people are followed to gain trendy information.*"

Interestingly, the learning cost in MM is relatively lower than that in EME. The observers can concurrently follow as many of the OSS project leaders as they want to expand their scope of knowledge. As stated by one interviewee (FG-1-C), "*I just follow those who have interesting projects in GitHub. There is no cost for me to do so, and I may get important information for my own projects from their updates and*

[5]FG-3-E denotes Focus Group 3, interviewee E.

activities." Theoretically, having observers does not cost the OSS project leaders. As expressed by one interviewee (FG-3-C), "*the following function in GitHub is extremely convenient and provides mutual benefits. We (observers) can easily get project-related information by following its leader, and that leader can also rely on (leveraging) us to promote her/his project.*" This comment provides descriptive explanations for research conjecture 2.

We gain additional interesting findings from the focus group interviews. As expected, code reuse is widely prevalent in developing an OSS project. Most interviewees agree with the following statement made by an interviewee (FG-2-E): "*I directly copy the codes from one project on which I work and paste them into another one (that I work for) …. No need to reinvent the wheel.*" This opinion is also echoed by another interviewee (FG-2-D): "*GitHub is like a bazaar where I can reuse the codes from other (developing) projects*." We find that the OSS participants hold a strong sharing mentality when they are subsequently asked to express their opinions in the activities of code reuse. An exemplary quote (from interviewee FG-3-D), "*The OSS is conducive to reducing the load of creating software from scratch. We can find needed codes for the existing projects, and we are not penalized for doing so*," adequately corroborates the assumption of the sharing mentality in OSS values and beliefs.

Table A5. Operationalization of the Constructs

Constructs and Sources	Items	Scale
Dependent Variable		
Intention to code reuse (Fishbein and Ajzen 1975; Sheppard et al. 1988)	(1) I intend to reuse the code learned from one OSS project leader in another OSS project. (2) I plan to continue reusing the code learned from one OSS project leader in another OSS project. (3) I plan to routinely reuse the code learned from one OSS project leader in another OSS project. (4) I predict that I will reuse the code learned from one OSS project leader in another OSS project.	Seven-point Likert scale (1 = strongly disagree; 4 = neutral; 7 = strongly agree)
Independent Variables		
EME (Self-developed)	(1) By following a particular person and contributing to her/his OSS project in GitHub, (2) I frequently learn from her/him. (3) The experience I learn from her/him is not valuable for me (reverse coded). (4) Co-working with her/him does not help me develop my experiences (reverse coded).	Seven-point Likert scale (1 = strongly disagree; 4 = neutral; 7 = strongly agree)
MM (Bandura 1977)	(1) I am able to enrich my knowledge by noting the information from the OSS project leaders I follow. (2) I have developed confidence in my ability to develop my own works by observing the information from the OSS project leaders I follow. (3) Gathering information from the OSS project leaders I follow has taught me how to develop my own works. (4) I have learned how to develop my own works better by monitoring the OSS project leaders I follow.	Seven-point Likert scale (1 = strongly disagree; 4 = neutral; 7 = strongly agree)

Table A5. Operationalization of the Constructs (Continued)

Constructs and Sources	Items	Scale
Control Variables		
Attentive observation (Yi and Davis 2003)	(1) I have paid close attention to the news or information on GitHub. (2) The news or information gets my attention. (3) I am able to take notice of the news or information on GitHub. (4) When using GitHub, I am absorbed by the news or information presented.	Seven-point Likert scale (1 = strongly disagree; 4 = neutral; 7= strongly agree)
OSS values (Stewart and Gosain 2006)	(1) I value sharing knowledge. (2) I believe in helping others. (3) I place great value on technical knowledge. (4) I am driven by a desire to learn new things. (5) I think cooperation is important. (6) I value the reputation gained from participating in open-source projects.	Seven-point Likert scale (1 = strongly disagree; 4 = neutral; 7= strongly agree)
OSS belief (Stewart and Gosain 2006)	(1) I believe that the best code wins out in the end. (2) I believe free software is better than commercial software. (3) I think information should be free. (4) I believe that with enough people working on a project, any bug can be quickly found and fixed. (5) I believe that you only become a hacker when others call you a hacker.	Seven-point Likert scale (1= strongly disagree; 4 = neutral; 7 = strongly agree)
Perceived importance of code reuse/sharing mentality/team diversity (Self-developed)	(1) Reusing code is important for OSS projects. (2) Sharing is the core in OSS. (3) Cooperating with stable people across OSS projects is preferred.	Seven-point Likert scale (1 = strongly disagree; 4 = neutral; 7 = strongly agree)
Tenure in GitHub	How long have you used GitHub?	0: Less than 1 year 1: 1-3 years 2: 3-5 years 3: More than 5 years
OSS usage experiences	How long have you used the OSS products?	0: Less than 1 year 1: 1-3 years 2: 3-5 years 3: 5-7 years 4: More than 7 years
Gender	Are you _____?	0: Female, 1: Male
Age	What is your age?	0: Younger than 20 years 1: 20 years to 24 years 2: 25 years to 29 years 3: 30 years to 34 years 4: 35 years to 39 years 5: 40 years to 44 years 6: 45 years to 49 years 7: 50 years and over

Table A6. Coding			
	Codename	**Meaning**	**Percentage**
First Topic: Understanding Open-Source Software			
Word count: 10,288 Code count: 223	1-FD	Free distribution, including the reuse and redistribution of source code	32%
	1-KS	Knowledge sharing	29%
	1-SM	Social movement	15%
	1-CS	Cost-free software	13%
	1-IP	Intellectual property and licenses	10%
Second Topic: Thoughts on Code Reuse			
Word count: 7,860 Code count: 168	2-SE	Save efforts (by reusing the existing code)	37.5%
	2-FC	Efficient software development completion	32%
	2-QI	Improved quality of the reused code	20%
	2-LB	Learning through reusing (existing code)	9%
Third Topic: Dynamics in the Follower–Leader Relationship			
Word count: 9,148 Code count: 198	3-DV	Diversity of collaborators	31%
	3-RC	Reduction in communication/coordination cost	28%
	3-SK	Seeking qualified and updated knowledge	27%
	3-AL	Acquiring legitimacy	13%

References

Au, Y. A., Carpenter, D., Chen, X., and Clark, J. G. 2009. "Virtual Organizational Learning in Open Source Software Development Projects," *Information & Management* (46:1), pp. 9-5.

Bandura, A. 1977. *Social Learning Theory*, Englewood Cliffs, NJ: Prentice Hall.

Cohen, J.B., and Reed, A. 2006. "A Multiple Pathway Anchoring and Adjustment (MPAA) Model of Attitude Generation and Recruitment," *Journal of Consumer Research*, (33:1), pp. 1-15.

Faraj, S., Kudaravalli, S., and Wasko, M. 2015. "Leading Collaboration in Online Communities," *MIS Quarterly* (39:2), pp. 393-411.

Fielding, R. T. 1999. "Shared Leadership in the Apache Project," *Communications of the ACM* (42:4), pp. 42-43.

Fishbein, M., and Ajzen, I. 1975. *Belief, Attitude, Intention and Behavior: An Introduction to Theory and Research*, Reading, MA: Addison-Wesley.

Fleming, L., and Waguespack, D. M. 2007. "Brokerage, Boundary Spanning, and Leadership in Open Innovation Communities," *Organization Science* (18:2), pp. 165-180.

Giuri, P., Rullani, F., and Torrisi, S. 2008. "Explaining Leadership in Virtual Teams: The Case of Open Source Software," *Information Economics and Policy* (20:4), pp. 305-315.

Ho, S. Y., and Richardson, A. 2013. "Trust and Distrust in Open Source Software Development," *The Journal of Computer Information Systems* (54:1), pp. 84-93.

Johnson, S. L., Safadi, H., and Faraj, S. 2015. "The Emergence of Online Community Leadership," *Information Systems Research* (26:1), pp. 165-187.

Lerner, J., and Tirole, J. 2002. "Some Simple Economics of Open Source," *The Journal of Industrial Economics* (50:2), pp. 197-234.

Li, Y., Tan, C. H., and Teo, H. H. 2012. "Leadership Characteristics and Developers' Motivation in Open Source Software Development," *Information & Management* (49:5), pp. 257-267.

O'Mahony, S., and Ferraro, F. 2007. "The Emergence of Governance in an Open Source Community," *Academy of Management Journal* (50:5), pp. 1079-1106.

McCulloch, C. E., and Neuhaus, J. M. 2001. *Generalized Linear Mixed Models*, Chichester, UK: John Wiley & Sons, Ltd.

Sen, R. 2007. "A Strategic Analysis of Competition Between Open Source and Proprietary Software," *Journal of Management Information Systems* (24:1), pp. 233-257.

Sheppard, B. H., Hartwick, J., and Warshaw, P. R. 1988. "The Theory of Reasoned Action: A Meta-Analysis of Past Research with Recommendations for Modifications and Future Research," *Journal of Consumer Research* (15:3), pp. 325-343.

Someren, M. V., Barnard, Y. F., and Sandberg, J. A. 1994. *The Think Aloud Method: A Practical Approach to Modeling Cognitive Processes*, Waltham, MA: Academic Press.

Stewart, K. J., and Gosain, S. 2006. "The Impact of Ideology on Effectiveness in Open Source Software Development Teams," *MIS Quarterly* (30:2), pp. 291-314.

Yi, M. Y., and Davis, F. D. 2003. "Developing and Validating an Observational Learning Model of Computer Software Training and Skill Acquisition," *Information Systems Research* (14:2), pp. 146-169.