Transactions on Replication Research

Conceptual Replication DOI: 10.17705/1atrr.00057 ISSN 2473-3458

Coordinating Interdependencies in an Open Source Software Project: A Replication of Lindberg, et al.

Randy V. Bradley
Department of Supply Chain Management, The University of Tennessee - Knoxville
rbradley@utk.edu

Audris Mockus
Department of Electrical Engineering and Computer Science, The University of Tennessee – Knoxville
audris@utk.edu

Yuxing Ma
Department of Electrical Engineering and Computer Science, The University of Tennessee – Knoxville
yma28@vols.utk.edu

Russell Zaretzki
Department of Business Analytics and Statistics, The University of Tennessee – Knoxville
rzaretzk@utk.edu

Bogdan C. Bichescu
Department of Business Analytics and Statistics, The University of Tennessee – Knoxville
bbichescu@utk.edu

Abstract:

The current study is a full replication (conceptual and empirical) of "Coordinating Interdependencies in Online Communities: A Study of an Open Source Software Project" Lindberg et al (2016), which addresses the question of how OSS communities address unresolved interdependencies. Following the original study, we analyze project development data, archived in the GitHub repository, for the OSS project Rubinius. The analysis explores relationships among development and developer interdependencies as well as activity and order variation. Further, we extend the original study by examining the core relationships in the original study and investigating the external generalizability of the results by replicating the analysis on three analogous OSS projects: JRuby, mruby, and RubyMotion. These offer an opportunity to evaluate the generalizability of the original study to projects of different sizes and amount of activity, yet similar otherwise to the project in the original study. Another extension is the use of an additional control variable, length of activity sequence, which proves to have substantial implications of the study's focal relationships. We find that three out of the four projects we analyze support the findings of the original study as it pertains to four relationships in the original study: order variation and developer interdependencies, activity variation and developer interdependencies, order variation and development interdependencies, and development and developer interdependencies. We also discuss the implications of our findings, especially in cases where the replication results differ from those in the original study and offer suggestions for future research that can help advance this stream of research.

Keywords: Open Source Software, Interdependencies, Coordination

The manuscript was received 01/14/2019 and was with the authors 11 months for 3 revisions.

1 Introduction

The question of how decentralized open source software (OSS) development communities manage technical (i.e. code that depends on components developed elsewhere) and social (i.e. where developer decisions affect others) interdependencies is both interesting and critical. As the Internet, e-commerce, and even consumer devices rely on the critical software developed by open source communities (Eghbal, 2016), the effective coordination of these interdependencies is truly an issue of societal importance. Failure can lead to quality problems (Cataldo, Mockus, Roberts, & Herbsleb, 2009) and delays (Herbsleb & Mockus, 2003). Avoiding these interdependency-induced issues may be accomplished through alignment of development activities with the dependencies (Cataldo, Herbsleb, & Carley, 2008).

In their original study, "Coordinating Interdependencies in Online Communities: A Study of an Open Source Software Project," (Lindberg et al., 2016) investigate how OSS communities address *unresolved interdependencies* both theoretically and through an empirical assessment of project development data archived in the GitHub repository. Through an exploratory case study of an online community around the OSS project Rubinius, they offer insights into how online communities manage both *unresolved developer and development interdependencies*. They identify *activity variation* and *order variation* as two key outcome measures associated with the aforementioned interdependencies.

The original study provides meaningful contributions for developers and researchers by proffering a theory for how developer routines and forms of variation serve as emergent coordination mechanisms in an OSS community. The findings indicate a significant difference between pull requests with *unresolved development interdependencies* and pull requests without such dependencies. More specifically, Lindberg et al. (2016) report that the relationship between *development interdependencies* and *activity (task) variation* is wedge shaped. That is to say their analysis reveals that *activity variation* increases as *unresolved development interdependencies* increase. However, low degrees of *development interdependencies* are associated with a large range of activity variation levels, while high degrees of *development interdependencies* are rarely observed only with high degrees of *activity variation*. Alternatively, there is no discernable relationship between *development interdependencies* and *order variation* (see Figure L1[1]). In contrast, *developer interdependencies* has a positive linear association with *activity variation*. In essence, the original study suggests that as more developers become involved with a particular pull request there is an increase in activity variation. With regards to order variation, the original study reports a wedge-shaped relationship between *developer interdependencies* and *order variation*, such that a large range of order variation is observed with low degrees of *unresolved developer interdependencies*, whereas little range of *order variation* is observed as *developer interdependencies* increase (see Figure L3). Lastly, the original study's findings suggest that as *development interdependencies* increase so does the degree of *developer interdependencies* (see Figure L2).

Despite the utility of the original study's process-based grounded theory approach to elucidating the relationships between both *unresolved developer and development interdependencies* and developer activities (i.e. *activity variation* and *order variation*), we believe that replication of empirical results through independent collection of the original sample from the same data archives, as well as additional open source ecosystems, is also of considerable worth to the research community. While reproducibility, vis-a-vis collection of a new sample in an appropriate sampling frame, is clearly critical to a healthy scientific ecosystem, Lindberg et al. (2016) is representative of many of the potential issues and challenges associated with the use of archival, observational data sources for social science. The novelty and relevance of the topic addressed by Lindberg et al. (2016) is likely to spawn a substantial amount of additional work in this domain. However, given the nuances and difficulties in executing such observational studies that rely on the extraction of archival data from OSS online communities or repositories, there is the need to ensure reproducibility (Nosek et al., 2015; Open Science Collaboration, 2015) of such original work deriving from complex big data, for example by sharing the data and source code and other approaches employed by open science (Boulton et al., 2012). In addition to adding scientific validity, the replication of observational big data studies increases assurances that (i) operationalizations of measures used in the original studies accurately reflect underlying concepts and (ii) the modeled relationships do not omit relevant latent variables.

[1] The letter proceeding each table and figure indicates whether it is based on the original study (L) or the replication study (R) as it pertains to the Rubinius project. Further, figures prefixed with JR, MR, and RM, refer to JRuby, mruby, and RubyMotion, respectively.

Lindberg et al. (2016) acknowledge that their contributions are limited by the scope of their inquiry and data. Specifically, they (i) allude to the potential existence of other aspects of software development that they could not observe that may influence coordination, and (ii) mention that their "measures are derived from pull requests as they appear after a merge or reject decision has been made". These two limitations suggest potentially unaccounted for latent variables and open the door for alternative operationalizations of the study's key variables in future work. Therefore, this replication study is interested in testing whether or not similar findings are identified after augmenting the original analysis with additional relevant variables and alternative operationalizations of the variables found to be associated with developer or development interdependencies.

To further assess the generalizability of the original study's findings, we analyze the relationships on a set of three additional projects that were selected based on similarities in functionality and contemporaneous time horizons to the original project. In summary, we find moderately strong support for the original study's findings. However, given that our findings of support differ across relationships with respect to the additional projects considered in the additional replications, we find less support for its generalizability in the context of online communities.

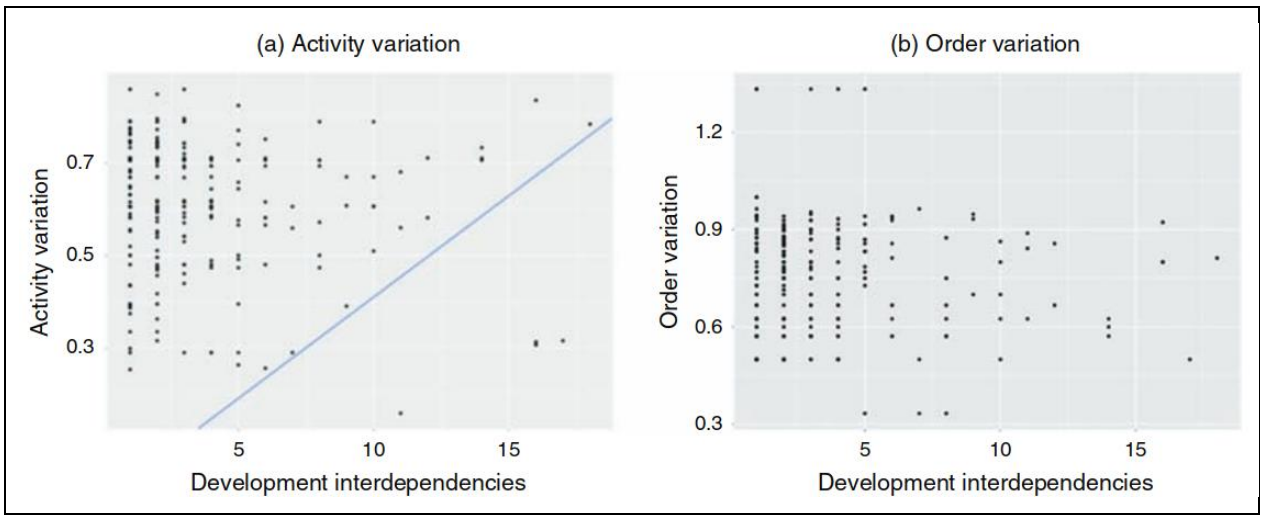


Figure L1. Routine Variation and Development Interdependencies

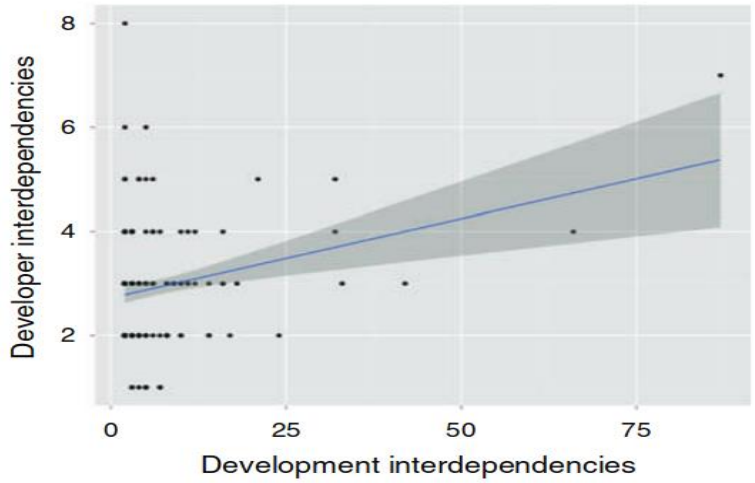


Figure L2. Relationship Between Development and Developer Interdependencies

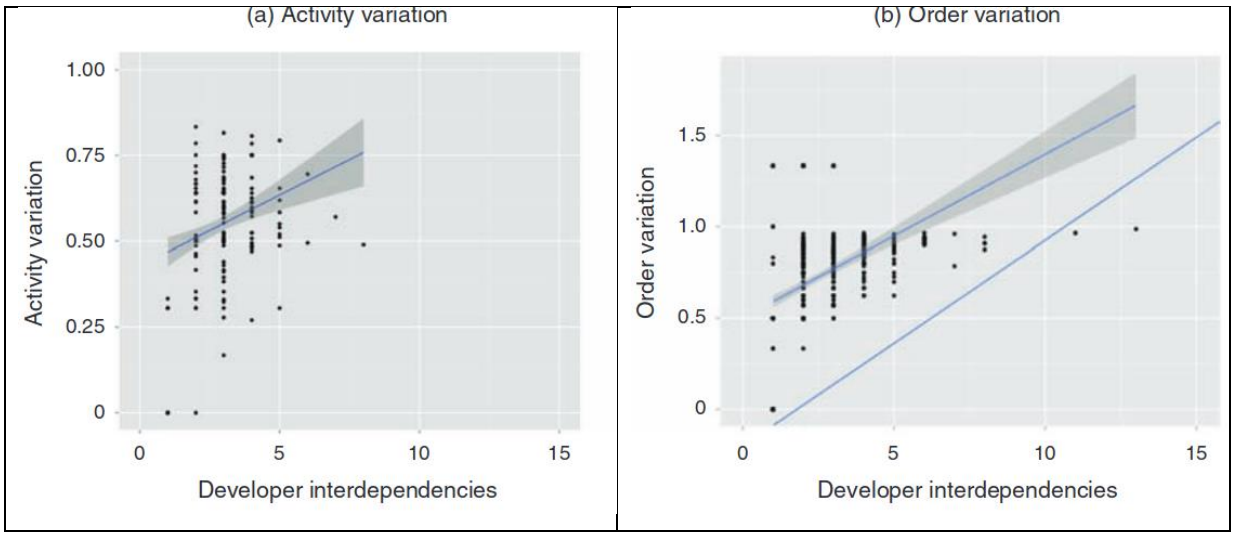


Figure L3. Routine Variation and Developer Interdependencies

2 Methods

We perform a full replication of Lindberg et al. (2016), analyzing *unresolved development and developer interdependencies* in relation to routine variation (i.e., *activity and order variation*). The types of replication studies may vary, for example, (Gómez, Juristo, & Vegas, 2010) define dozens of names for different replication types. To be specific, by full replication we mean that we perform a conceptual and empirical replication. By conceptual replication we test the generalizability of the same ideas to different populations. By empirical replication we seek to confirm the previous findings using a different set of specific methods that test the same idea where we modify models and introduce additional control variables.

As a full replication, we examine the same variables used in the original study, but also augment the analysis by including an additional variable not considered previously. We add *length of activity sequence* (i.e., the number of activities in a pull request) to our analysis. *Length of activity sequence* is included because the distribution of activity variation and order variation functionally depends on the number of activities in that sequence and is necessary to distinguish the deterministic aspect of the relationship from the phenomena of interest. We also investigate the same relationships (i.e. *unresolved development and developer interdependencies* in relation to activity and order variation) that made up the quantitative analysis in the original study.

Similar to the original study, we leverage data from GitHub (via the GitHub API) for the original OSS project, Rubinius, and for three additional Ruby-language projects including JRuby, mruby, RubyMotion, that, like Rubinius, are all compilers. The data on events, issues, issue comments, pull requests, pull request comments, commits, and pull request files were obtained in September 2019. Except where noted in the paper, we adhere to the data retrieval methods described in the original study. In instances where the original study lacks sufficient detail regarding aspects of data collection and aggregation to replicate the process, we follow the overall method of collecting archival data as outlined in the mining software repositories (MSR) literature (see Mockus, Fielding, and Herbsleb (2002) for an example). In the remaining instances, where there isn't agreed upon guidance in the MSR literature, we experiment with several plausible approaches and select the one that produces results that most closely resembles the data set described in the original study. For example, the original study states, "we collected this data over a 12-month period (January 6, 2012, to January 6, 2013)." It is not clear, however, whether only issues *created* during this time period are selected, or whether the data set also includes issues that *overlap* with this period (e.g., issue created before January 6, 2012 but resolved after January 6, 2013, or issues created before January 6, 2012 but resolved between January 6, 2012 and January 6, 2013). We ultimately settle on the choice/approach that yields counts of pull requests and issues that are closest to the values in the original study (see Table R1 for a comparison).

To test the generalizability of the findings in the original study, we further extend the replication study to three other Ruby compiler projects (JRuby, mruby and RubyMotion). Among other things, this allows us to see if the project size affects the relationships discussed in the original study. According to Table R1, JRuby is about twice the size (as measured by the number of issues) of Rubinius and mruby, whereas RubyMotion is approximately an order of magnitude smaller than all of the other projects.

All data for all projects are filtered from a single time period defined by the original study (January 2012 – January 2013) and then the constructs of interest (*activity and order variation*) and *interdependencies (development and developer)* are calculated. Finally, linear regression models are fit to estimate relations between the five pairs of variables shown in Figures R1, R2, R3. JRuby is an exception with few activities recorded during the interval used in the original study. We therefore shift the study period for JRuby to a later date range (September 2014 – September 2015), during which activity is sufficiently intensive to obtain all measures of interest.

2.1 Analytical Approach

The analysis in this study uses linear regression[2] to test the significance of the relationships between the response measures representing coordination routines (*activity variation* and *order variation*) and explanatory variables (log (*development interdependencies)* and *developer interdependencies*). In the case of *order variation* as a response, a separate simple linear regression is fit using both explanatory variables. In the case of *activity variation*, two versions of this relationship are estimated and evaluated for both log (*development interdependencies)* and *developer interdependencies*:

1. A simple linear regression of *activity variation* vs. both *interdependencies*,
2. a multiple linear regression of *activity variation* vs. each of the *interdependencies* with the *length of activity sequence*; definition given above in the Methods section.

For each model, we report the coefficient, *p*-value, and adjusted R^2 value of the model.

2.2 Assumptions and Methodological Differences

The slight differences observed in Table R1 in the numbers/counts of activities may be due to assumptions made in the replication study or deviations in the methodological approach for collecting the data. Additionally, we use the log transformation of *development interdependencies* to satisfy model assumptions. A further reason for the numerical discrepancies could be due to the fact that the underlying data on GitHub changes over time, (e.g., some issues may get resolved, others may become pull requests). Since we obtain data from GitHub at a later date than the original study, it is possible that some issues may have a different activity classification during our collection period as compared to the original study.

[2] In the original study, beta coefficients are presented but the study does not explicitly state what methodology was used to obtain the beta coefficients. However, we inferred that a linear regression analysis would be appropriate primarily because the nature of the figures and variables suggest that a linear model was employed.

Table R1. Comparison of issue and pull request counts used to identify criteria for inclusion

Quantity	Rubinius(Original Study)	Rubinius (Replication)	JRuby	mruby	RubyMotion
Pull requests	198[3]	267	405	479	34
Issues	488[4]	356	960	215	1
Activities: Assigned	3	2	230	1	0
Activities: Closed	857	571	1030	678	23
Activities: Commented	1134	1120	3255	569	0
Activities: Mentioned	440	443	1358	179	22
Activities: Merged	369	191	315	387	19
Activities: Opened	268	261	405	479	34
Activities: Referenced	470	455	919	670	22
Activities: Reopened	17	14	62	12	0
Activities: Reviewed	146	182	500	3	10

2.2.1 Pull Requests and Issues

As in the original study, the basic analysis unit in the replication study is a pull request. Whereas the original study defines issues as pull requests, GitHub refers to issues as a superset of pull requests (i.e., all pull requests are issues, but not all issues are pull-requests). In the replication study, we, therefore, consider issues with code attachments to be pull requests and issues without code attachments to be simply issues.

2.2.2 Activities

The original study uses the term activity, which is a term not native to GitHub. We assume that “activity” corresponds to GitHub's “issue events”, because the types of issue events recognized on GitHub are nearly the same as the types of activities in Table R1 of the original study. Based on this assumption, we extract all events associated with an issue and treat them as synonymous to activities, and, thus count the number of times each issue event type appears.

3 Results

3.1 Activity Variation and Development Interdependencies

The original study finds a significant relationship between *development interdependencies* (X) and *activity variation* (Y) using a delta test[5] but without assessing the slope of the relationship. However, using a more standard regression approach, we do not confirm this finding through replication of the Rubinius study, nor for the other three projects used in our replication. See Table R2[6] for comparisons of findings between the original Rubinius project and the Rubinius replication project; in this case, neither the simple linear regression model (*activity variation* vs. *development interdependencies*) nor the model that adds *length of activity sequence* are significant. For each relationship, Tables R3,R4, and R5 contains comparative findings from the original Rubinius project with methodological replications for the JRuby, mruby, and RubyMotion projects, respectively (see Appendices B and BC for full regression results). While some relationships show significance, the direction of significance is opposite to the original study as judged through Figure L1, which suggests a lack of support. Both tables contain the beta coefficient, statistical

[3] The original study refers to these as pull requests with code attached.
[4] The original study refers to these as pull requests without code attached.
[5] This finding uses a methodology called a delta test that is not replicated in our study in lieu of more standard regression methods. The “delta” test proposed by (Bardsley, Jorgensen, Alpert, & Ben-Gai, 1999), is a statistical procedure to test for spatial homogeneity (empty areas) in scatter plots. Bardsley et al. (1999) claim that this might be an appropriate way to evaluate regression type relationships when the relationships (correlations) are weak. While the methodology does make sense in some cases, we find the approach to be highly problematic in the context of our replication. Application of this method hinges on the fact that, when the data is created, certain (x, y) pairs may be much rarer than others leading to few points in certain zones. For example, developer interdependencies (x) is operationalized by simply counting the number of developers that work on a pull request. Naturally, large values tend to be uncommon. It’s extremely rare to find a pull request that is worked on by more than 10 developers, irrespective of activity or order variation values. In fact, we only found 2 out of 623. Therefore, the right half of the plot will tend to be relatively empty, making it inevitable that the test would reject the hypothesis of random scatter. Of course, this is due to sampling bias and does not mean that order variation increases with developer interdependencies. In general, sampling should be uniform across X when using the delta test, which is not the case here, suggesting that the test is inappropriate in this replication study.

[6] All replication model coefficients presented in this and subsequent tables are unstandardized, whereas it is plausible that the coefficients in the original study are standardized. This might explain the differences in coefficient magnitudes across the two studies, but in terms of statistical significance of variables, this doesn’t inhibit comparison across studies.

significance of the fitted coefficients of a linear regression on routine variations vs interdependencies, and the adjusted R^2. Each row in Tables R2-R5 represents a separate regression. We include two rows for each *activity variation* related relationship: the first row shows the initial result after applying the simple linear regression as in the original paper, while the second row shows the result after taking *length of activity sequence* into consideration. Within each table, *Coeff* represents the coefficient of interest and *Adj. R^2* is the adjusted R-squared statistic (which we report as a decimal (0-1), instead of a percentage (0-100)) for each model we investigate. The column *Result* shows the estimated coefficient and whether or not it is statistically significant. If the coefficient is not significant we affix the *NS* label, whereas we affix the *NS** label to a relationship that is significant but in the opposite direction of what the original study reports. Further, results in bold indicate findings consistent with the original study. Although the adjusted R^2 statistic is not referenced in the original work, it is a standard tool in assessing the predictive strength of evidence when linear regression analysis is used.

3.2 Order Variation and Development Interdependencies

Using the delta test, the original study finds no discernible relationship between *development interdependencies* (X) and *order variation* (Y) and our findings based on linear regression are consistent with that. We are not able to find significant relationships in the Rubinius project, nor in two of the other replication projects. However, we are able to detect a significant relationship in the JRuby project, but the coefficient has a sign opposite of that in the original study, also indicating a lack of support.

3.3 Activity Variation and Developer Interdependencies

The original study finds a positive and significant relationship between *developer interdependencies* (X) and *activity variation* (Y). Consistent with the original study, we confirm that finding via replication of the Rubinius in the simple linear regression model, while the model that adds the *length of activity sequence* control is no longer significant. In the case of JRuby, mruby and RubyMotion, Tables R3-R5 show inconsistent results across the projects. In no cases is the model augmented by *length of activity sequence* significant. Conversely, we find a significant, but negative relationship in the mruby project while JRuby and RubyMotion have significant positive associations when no control was included.

3.4 Order Variation and Developer Interdependencies

The original study finds a positive and significant relationship between *developer interdependencies* (X) and *order variation* (Y). Our results confirm that finding via replication of the Rubinius, JRuby, and RubyMotion projects. Conversely, we find a significant, but negative relationship in the mruby project resulting in moderately strong support for the generalizability of the original study results.

Table R2. Rubinius (Original) vs. Rubinius (Replication) Results

Relationship/Project	Rubinius - Original			Rubinius - Replication			
	Result	Coeff	*p*	Result	Coeff	*p*	Adj. R^2
Development Interdependencies/ Activity Variation	Sup[7]	Not Reported	Not Reported	NS	0.0049	0.403	< 0
Development Interdependencies/ Activity Variation with *Length of Activity Sequence*	Not Applicable	Not Applicable	Not Applicable	NS	0.0010	0.843	0.24
Development Interdependencies/ Order Variation	NS	Not Reported	Not Reported (for linear regression)	**NS**	-0.0042	0.499	<0
Developer Interdependencies/ Activity Variation	Sup	2.08	<0.001	**Sup**	0.1121	<0.001	0.06
Developer Interdependencies/ Activity Variation with *Length of Activity Sequence*	Not Applicable	Not Applicable	Not Applicable	NS	0.0146	0.061	0.07
Developer Interdependencies/	Sup	1.16	<0.001	**Sup**	0.0443	<0.001	0.02

[7] Lindberg et al (2016) do not report linear regression coefficients or p-values for development interdependencies, instead they use a separate test that was not replicated; see discussion in Section 3.

Table R2. Rubinius (Original) vs. Rubinius (Replication) Results							
Relationship/Project	Rubinius - Original			Rubinius - Replication			
	Result	Coeff	*p*	Result	Coeff	*p*	Adj. R^2
Order Variation							
Development Interdependencies/ Developer Interdependencies	Sup	1.59	<0.001	**Sup**	0.0297	0.03	0.01

3.5 Development and Developer Interdependencies

The original study finds a positive and significant relationship between *development and developer interdependencies*. Consistent with the original study, we find support in our replication of the Rubinius project. Although the RubyMotion project does not show a significant relationship, we find agreement with the relationship in JRuby and mruby, which offers moderately strong support for the generalizability of the original study's results.

Table R3. Rubinius (Original) v. JRuby Results							
Relationship/Project	Rubinius - Original			JRuby			
	Result	Coeff	*p*	Result	Coeff	*p*	Adj. R^2
Development Interdependencies/ Activity Variation	Sup	Not Reported	Not Reported	NS	-0.0062	0.152	<0.01
Development Interdependencies/ Activity Variation with *Length of Activity Sequence*	Not Applicable	Not Applicable	Not Applicable	NS	-0.0084	0.081	0.003
Development Interdependencies/ Order Variation	NS	Not Reported	Not Reported (for linear regression)	NS*	-0.032	<0.001	0.10
Developer Interdependencies/ Activity Variation	Sup	2.08	<0.001	**Sup**	0.0489	<0.001	0.05
Developer Interdependencies/ Activity Variation with *Length of Activity Sequence*	Not Applicable	Not Applicable	Not Applicable	NS	0.0117	0.098	0.10
Developer Interdependencies/ Order Variation	Sup	1.16	<0.001	**Sup**	0.0433	<0.001	0.03
Development Interdependencies/ Developer Interdependencies	Sup	1.59	<0.001	**Sup**	0.1688	<0.001	0.09

Table R4. Rubinius (Original) v. mruby Results							
Relationship/Project	Rubinius - Original			mruby			
	Result	Coeff	*p*	Result	Coeff	*p*	Adj. R^2
Development Interdependencies/ Activity Variation	Sup	Not Reported	Not Reported	NS	-0.0041	0.379	<0.01
Development Interdependencies/ Activity Variation with *Length of Activity Sequence*	Not Applicable	Not Applicable	Not Applicable	NS	-0.0047	0.195	<0.01
Development Interdependencies/ Order Variation	NS	Not Reported	Not Reported (for linear regression)	**NS**	-0.0066	0.163	<0.01
Developer Interdependencies/ Activity Variation	Sup	2.08	<0.001	**NS***	-0.0172	0.002	<0.01
Developer Interdependencies/ Activity Variation with *Length of Activity Sequence*	Not Applicable	Not Applicable	Not Applicable	NS*	-0.0484	<0.001	0.09
Developer Interdependencies/ Order Variation	Sup	1.16	<0.001	NS*	-0.03	0.004	0.02
Development Interdependencies/ Developer Interdependencies	Sup	1.59	<0.001	**Sup**	0.101	<0.001	0.03

Table R5. Rubinius (Original) v. RubyMotion Results							
Relationship/Project	Rubinius - Original			RubyMotion			
	Result	Coeff	*p*	Result	Coeff	*p*	Adj. R^2
Development Interdependencies/ Activity Variation	Sup	Not Reported	Not Reported	NS	-0.0658	0.379	<0
Development Interdependencies/ Activity Variation with *Length of Activity Sequence*	Not Applicable	Not Applicable	Not Applicable	NS*	-0.1188	0.011	0.65
Development Interdependencies/ Order Variation	NS	Not Reported	Not Reported (for linear regression)	**NS**	-0.0928	0.163	0.03
Developer Interdependencies/ Activity Variation	Sup	2.08	<0.001	**Sup**	0.2237	0.002	0.24
Developer Interdependencies/ Activity Variation with *Length of Activity Sequence*	Not Applicable	Not Applicable	Not Applicable	NS	0.0165	0.80	0.58
Developer Interdependencies/ Order Variation	Sup	1.16	<0.001	**Sup**	0.1915	0.004	0.21
Development Interdependencies/ Developer Interdependencies	Sup	1.59	<0.001	NS	0.0564	0.748	<0

Figures R1, R2, and R3 plot the replicated Rubinius project data overlaying the simple linear regression fit. The corresponding illustrations for the JRuby (JR), mruby (MR) and RubyMotion (RM) projects can be found in Appendix A.

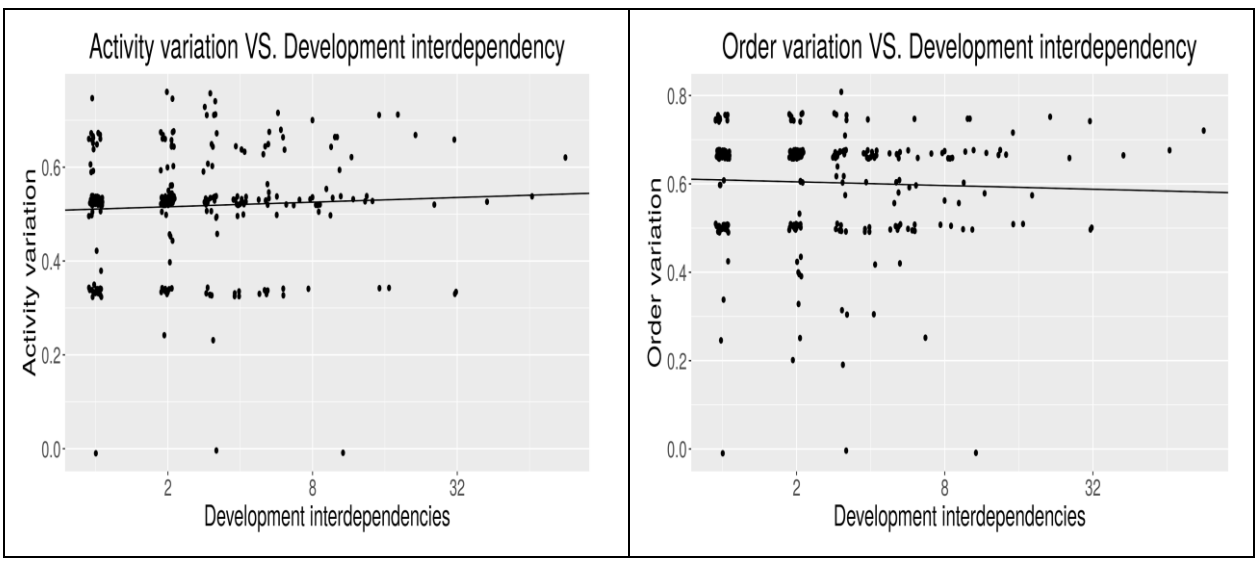


Figure R1: Routine Variation and Development Interdependencies

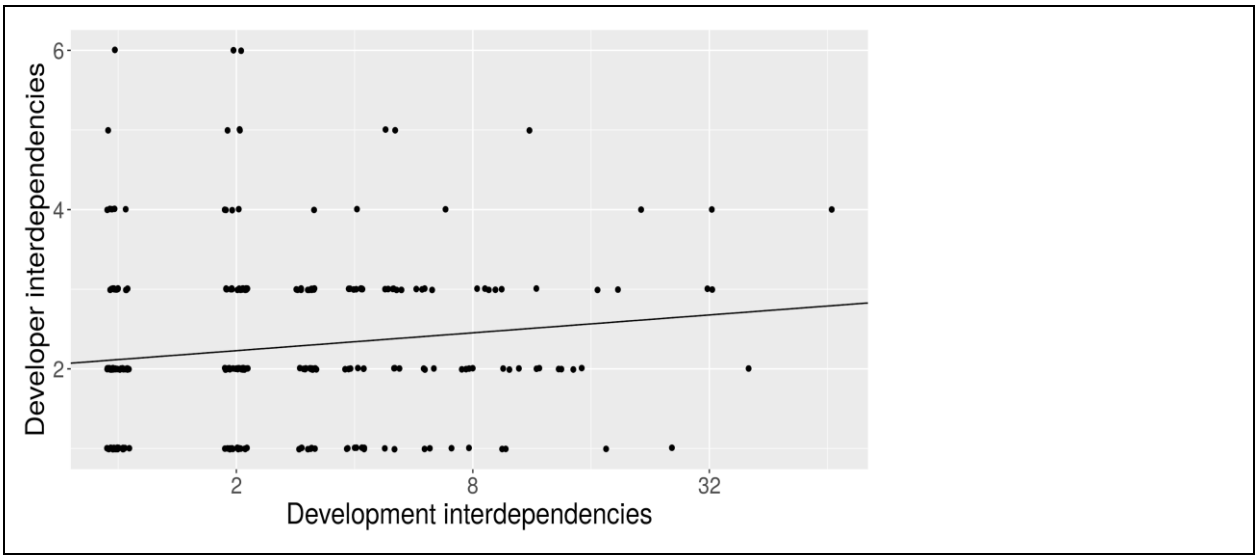


Figure R2: Relationship Between Development and Developer Interdependencies

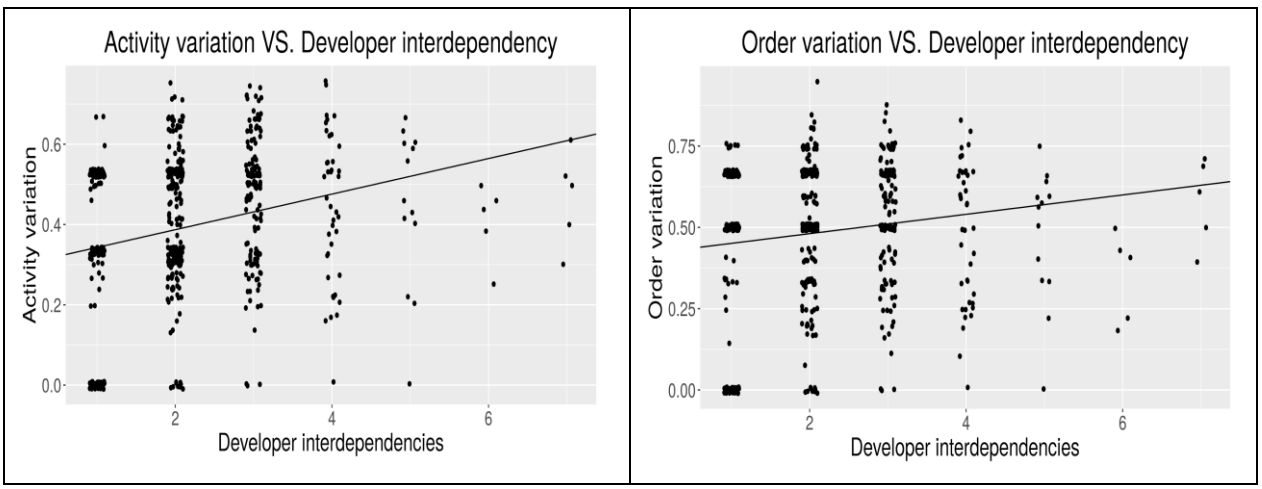


Figure R3: Routine Variation and Developer Interdependencies

4 Discussion

Below we discuss findings in the current replication along with limitations and offer suggestions for the direction of future work. Tests of the models estimated in Section 3 provide relatively conclusive evidence of our ability to replicate the results/corroborate the findings of Lindberg et al. (2016). Nevertheless, there are some noticeable differences between the findings that are worthy of further discussion.

The original study and this replication report on wedge-shaped relationships and linear relationships, respectively, as it pertains to the relationship between interdependencies and routine variation. From an interpretation standpoint, the original study's wedge-shaped analysis suggests that in certain cases there exists a relationship between interdependencies and routine variation, but the directionality and quantification/strength of that relationship is opaque. In contrast, the results of the replication study, which employs a linear model, suggests there exists a directional relationship between interdependencies and routine variation and quantifies (indicates the strength) the relationship. So, although both approaches can determine the presence (or absence) of a relationship, the linear model provides greater specificity as to the nature of the relationship. In the subsequent sections, we discuss the implications of the replication study's findings in relation to the original study's findings.

4.1 Activity Variation and Development Interdependencies

Given that the original study and this replication analyze this relationship from a different perspective (for reasons explained in the Results section), it is reasonable that the findings are not consistent. However, despite this point, our results are consistent with each other across the four projects examined in the replication, wherein we find no association between *activity variation* and *development interdependencies*. Additional insight can be gained by considering Figure R1 (left), wherein the data points don't appear to show a clear trend.

While the *development interdependencies* does not appear, on replication, to have a strong effect, including *length of activity sequence* shows a significant, but negative, relationship with activity variation, but only in the case of RubyMotion. When we adjust for the *length of activity sequence* as shown in Tables R2 and R3, the relationship is either no longer significant or becomes significant but in the opposite direction. There are a couple of reasons for the differences in the findings across projects. One reason could be that *length of activity sequence* might be a better explanatory variable than *development interdependencies*, with respect to *activity variation*. For example, previous studies suggest length of activity sequence is a primary determinant of pull request latency (Yu, Wang, Filkov, Devanbu, & Vasilescu, 2015; Yu, Wang, Yin, & Wang, 2016). A second explanation might be that the differences across projects are due to project sizes, where smaller projects may not have or need more sophisticated coordination routines (Blincoe, Valetto, & Damian, 2013). Yet another plausible explanation is the interplay between the *length of activity sequence* and *development interdependencies*. The synergy between the two might yield different results for *activity variation* than when considered individually. As such, future studies should consider whether the relationship between *activity variation* and *development interdependencies* may be moderated by *length of activity sequence,* which accounts for certain aspects of task complexity. If the coordination routines in more complex development tasks need to be different than for simpler development tasks, their association with *development interdependencies* would differ as well.

4.2 Activity Variation and Developer Interdependencies

With respect to this relationship, we confirm that in three of four projects analyzed, there exists a significant relationship between *activity variation* and *developer interdependencies*[8]. Thereby suggesting relative strong support for the findings in the original study. However, it should also be noted that when we control for l*ength of activity sequence*, we cannot confirm the existence of the focal relationship in any of the four projects. Again, we see that *length of activity sequence* acts as a suppressor and a more influential explanatory variable. For example, as illustrated by the substantially larger adjusted R^2 values for the models (across all projects) in which we control for *length of activity sequence*. One potential reason for this suppression effect is that longer *activity sequences* are typically associated with development tasks that are unusual (Gousios, Pinzger, & Deursen, 2014; Yu et al., 2015; Yu et al., 2016), thus making coordination more challenging. Furthermore, according to Gousios et al. (2014, p. 351), "the [pull request] discussion is usually brief: 95% of pull requests receive 12 comments or less (80% less than 4 comments). Similarly, the number of participants in the discussion is also low (95% of pull requests are discussed by less than 4 people). The number of comments in the discussion [the length of activity sequence] is moderately correlated with the time to merge a pull request ($\rho = 0.48$, $n = 141, 468$)."

4.3 Order Variation and Developer Interdependencies

Apart from mruby, the replication projects confirm the findings of the original study. One explanation for the differences for this particular project might relate to a larger number of open activities. Such open activities may represent pull requests with no discussion, thus lacking *order variation*.

4.4 Development Interdependencies and Developer Interdependencies

With respect to this relationship, apart from RubyMotion, three of four projects analyzed support the original study's finding that there exists a relationship between development and developer interdependencies. It should be noted that RubyMotion is the smallest of the projects we investigate, and one explanation for the differences for this particular project might relate to the lack of need to develop complex coordination routines for such a relatively small development project. Despite the finding differences in the RubyMotion project, we find there to be moderately strong support for the theoretical position of Lindberg et al. (2016).

[8]In the case of mruby, the relationship was not confirmed.

Additionally, from a practical standpoint the finding has merit since tasks involving more developers are more likely to require changes to more source code files and conversely.

4.5 Limitations and Future Research

Despite our attempt to replicate the results on the same project and on new projects, it is possible that the relationships that we find to be replicable might not generalize beyond the set of projects we consider. In the case of this replication, all projects investigated are compilers for the Ruby programming language. A potential extension that can be offered by future studies is to investigate research that investigates these relationships on projects that are dissimilar to the ones investigated by this replication study. It might be that either practices of Ruby developers or, more generally, of compiler developers, might not exhibit the patterns of activity routines as those associated with non-complier projects. Such studies could provide even greater insights as to the true nature of might find a stronger empirical support for these theoretical relationships promoted in this replication and the original study.

We point out a potential control variable that appears to impact some of the findings. Inclusion of additional control variables may need to be considered in the future research. It may be the case that the postulated relationships are mediated by additional latent variables that were not included in the models. For example, the size of the task, may influence our predictor in a way that makes it difficult to discern the effects of the other predictors. Future investigations are needed to ascertain the possibility and impact of such effects.

As discussed in the Methods section, even the archival data changes over time, hence we provide the exact state of the archival data we used in conjunction with the analysis scripts to facilitate future replications. Also, from a methodological perspective, some of the analysis methods may contain a subjective element, for example, the choice of the modeling technique, the identification of outliers, or other decisions on whether or not the modeling assumptions are satisfied. To mitigate this limitation, we try to explicitly state and justify all of these choices. Some of our analysis is conducted using a slightly modified methodology to ensure that the assumptions of the methods used to analyze the data are satisfied. It would be prudent for future research to take the peculiarities of the software data into account and to ensure that the assumptions of traditional methods, when applied on such data, are satisfied.

It is insightful to also consider the adjusted R^2 in the interpretation of Table R2 and Table R3. First, note the very low value of the adjusted R^2 for our replication models. For example, all simple linear fits of *activity variation* vs *development interdependencies* have a fit below 0.01, which indicates that the models do not explain the observed variance of the *activity variation* and that it is primarily driven by other factors that are not included in the model. This is corroborated by the increase in adjusted R^2 when *length of activity sequence* is taken into account. In general, these low values call into question the reproducibility of the relations studied and point to a range of exciting opportunities for future research. For example, a new and different perspective for studying the relationships between coordination needs and routines used to manage them in software development projects should be considered. Alternative approaches to operationalizing interdependencies and patterns of coding practices should be developed. Such developments will provide an important contribution that moves this area forward.

Another avenue for future research may be pursued to investigate the possibility that the operationalizations of the concepts used in this study may not adequately capture the totality of the underlying concepts and suggest revised, more comprehensive measures.

5 Conclusions

This research replicates the study presented by Lindberg et al. (2016) on the relationships between *developer* and *development interdependencies* and *activity* and *order routine variations* in OSS communities. In fact, the current work is a full replication (conceptual and empirical) and extension, in that it examines the core relationships in the original study across several projects of different sizes that, while matching in focus, provide some variation in the amount of project activity.

We find that three out of the four projects we analyze support the following three relationships from the original study:

1. order variation and developer interdependencies,
2. activity variation and developer interdependencies, and
3. development and developer interdependencies.

Further, the original study finds no significant relationship between *order variation* and *development interdependencies*, and we are able to also replicate this result across three of four projects in this study. However, in contrast to the original study, neither of the four projects we investigate in the replication study reveal a significant relationship between *activity variation* and *development interdependencies.* Overall, this replication study offers moderately strong support for the validity of the original study's findings, yet less support for its generalizability in the context of online communities.

Acknowledgments

This research was supported by NSF award IIS-1633437.

References

Bardsley, W., Jorgensen, M., Alpert, P., & Ben-Gai, T. (1999). A significance test for empty corners in scatter diagrams. *Journal of Hydrology, 219*(1-2), 1-6.

Blincoe, K., Valetto, G., & Damian, D. (2013). Do all task dependencies require coordination? the role of task properties in identifying critical coordination needs in software projects. *Proceedings of the 9th Joint Meeting on Foundations of Software Engineering*.

Boulton, G., Campbell, P., Collins, B., Elias, P., Hall, W., Laurie, G., O'Neill, O., Rawlins, M., Thornton, J., Vallance, P., & Walport, M. (2012). Science as an open enterprise. London: *The Royal Society*.

Cataldo, M., Herbsleb, J. D., & Carley, K. M. (2008). Socio-technical congruence: A framework for assessing the impact of technical and work dependencies on software development productivity. Paper presented at the *Second ACM-IEEE International Symposium on Empirical Software Engineering and Measurement*.

Cataldo, M., Mockus, A., Roberts, J. A., & Herbsleb, J. D. (2009). Software dependencies, work dependencies, and their impact on failures. *IEEE Transactions on Software Engineering, 35*(6), 864-878.

Eghbal, N. (2016). *Roads and bridges: The unseen labor behind our digital infrastructure*. Retrieved from https://www.fordfoundation.org/work/learning/research-reports/roads-and-bridges-the-unseen-labor-behind-our-digital-infrastructure/.

Gómez, O. S., Juristo, N., & Vegas, S. (2010). *Replications types in experimental disciplines.* Paper presented at the Proceedings of the *2010 ACM-IEEE International Symposium on Empirical Software Engineering and Measurement*.

Gousios, G., Pinzger, M., & Deursen, A. v. (2014). An exploratory study of the pull-based software development model. *Proceedings of the 36th International Conference on Software Engineering*.

Herbsleb, J. D., & Mockus, A. (2003). Formulation and preliminary test of an empirical theory of coordination in software engineering. Paper presented at the *ACM SIGSOFT Software Engineering Notes*.

Lindberg, A., Berente, N., Gaskin, J., & Lyytinen, K. (2016). Coordinating interdependencies in online communities: A study of an open source software project. *Information Systems Research, 27*(4), 751-772.

Mockus, A., Fielding, R. T., & Herbsleb, J. D. (2002). Two case studies of open source software development: Apache and Mozilla. *ACM Transactions on Software Engineering and Methodology (TOSEM), 11*(3), 309-346.

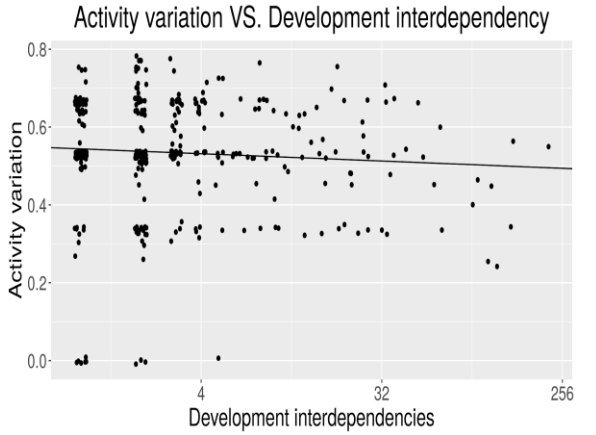
Nosek, B. A., Alter, G., Banks, G. C., Borsboom, D., Bowman, S. D., Breckler, S. J., Buck, S., Chambers, C. D., Chin, G., Christensen, G., Contesabile, M., Dafoe, A, Eich, E., Freese, J., Glennerster, R., Goroff, D., Green, D. P., Hesse, B., Humphreys, M., Ishiyama, J., Karlan, D., Kraut, A., Lupia, A., Mabry, P., Madon, T., Malhotra, N., Mayo-Wilson, E., McNutt, M., Miguel, E., Paluck, E. L., Simonsohn, U., Soderberg, C., Spellman, B. A., Turitto, J., VandenBos, G., Vazire, S., Wagenmakers, E., J., Wilson, R., & Yarkoni, T. (2015). Promoting an open research culture. *Science, 348*(6242), 1422-1425.

Open Science Collaboration. (2015). Estimating the reproducibility of psychological science. *Science, 349*(6251), aac4716.

Yu, Y., Wang, H., Filkov, V., Devanbu, P., & Vasilescu, B. (2015). Wait for it: Determinants of pull request evaluation latency on GitHub. Paper presented at the *2015 IEEE/ACM 12th Working Conference on Mining Software Repositories*.

Yu, Y., Wang, H., Yin, G., & Wang, T. (2016). Reviewer recommendation for pull-requests in GitHub: What can we learn from code review and bug assignment? *Information and software technology, 74*, 204-218.

Appendix A: Routine Variation and Interdependencies Figures for JRuby(JR), mruby(MR) and RubyMotion(RM)



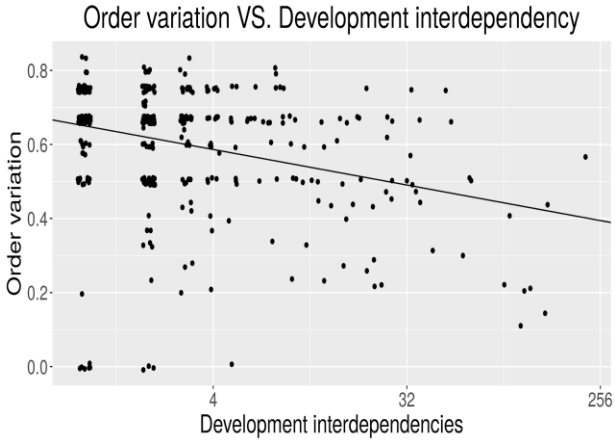


Figure JR. 1: Routine Variation and Development Interdependencies

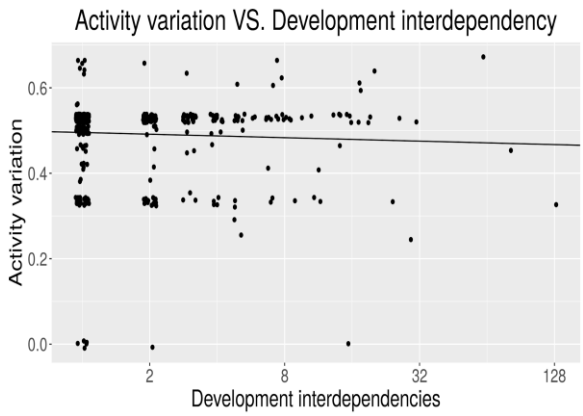


Figure MR. 1: Routine Variation and Development Interdependencies

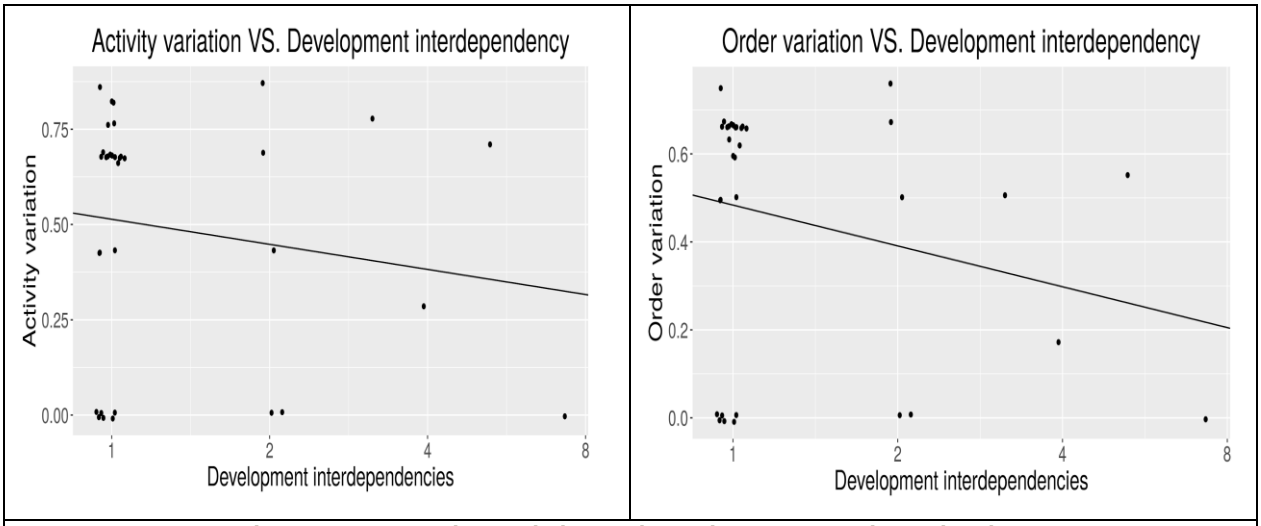


Figure RM. 1: Routine Variation and Development Interdependencies

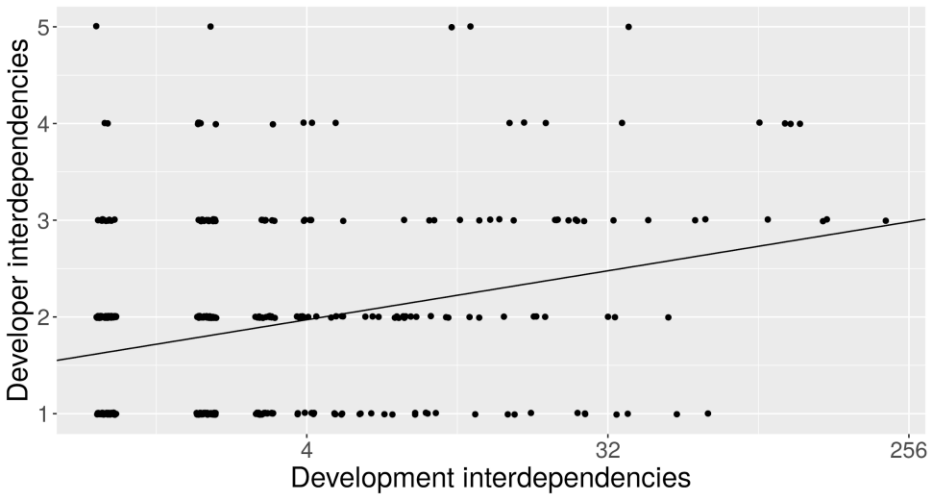


Figure JR. 2: Relationship Between Development and Developer Interdependencies

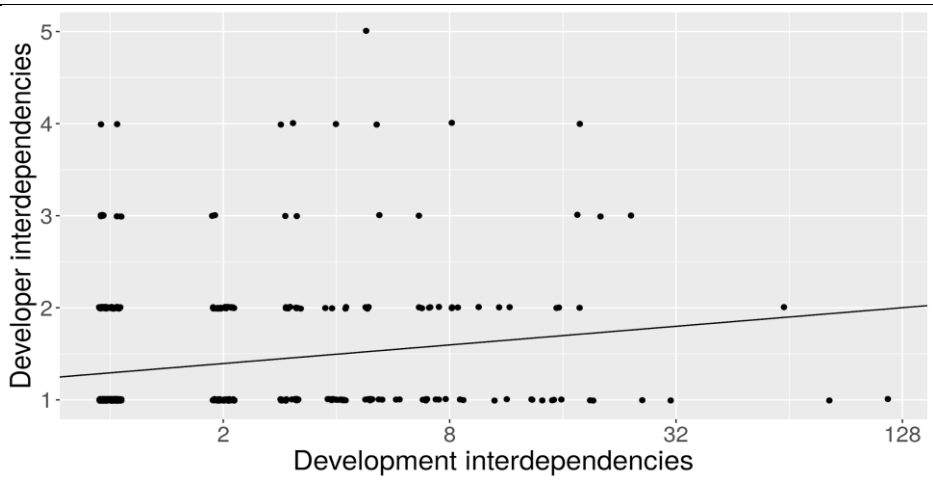


Figure MR. 2: Relationship Between Development and Developer Interdependencies

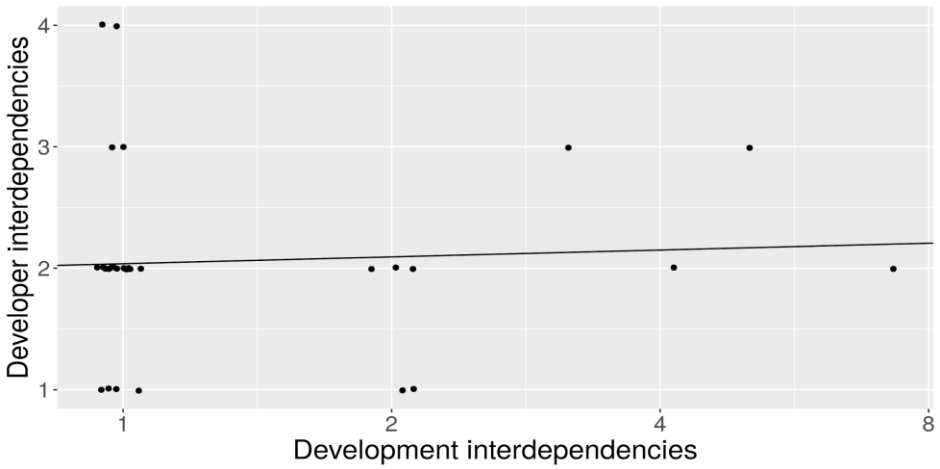


Figure RM. 2: Relationship Between Development and Developer Interdependencies

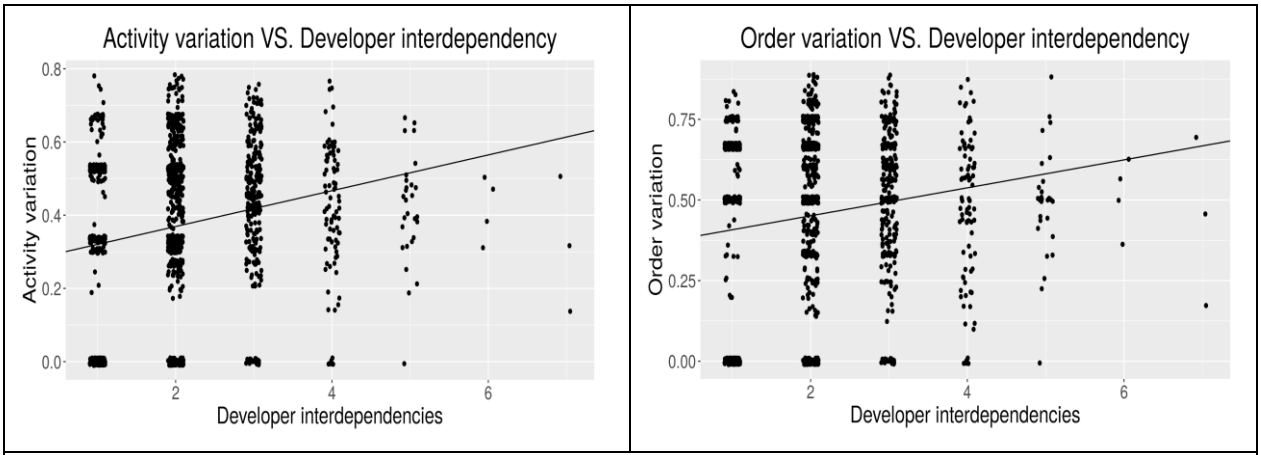


Figure JR. 3: Routine Variation and Developer Interdependencies

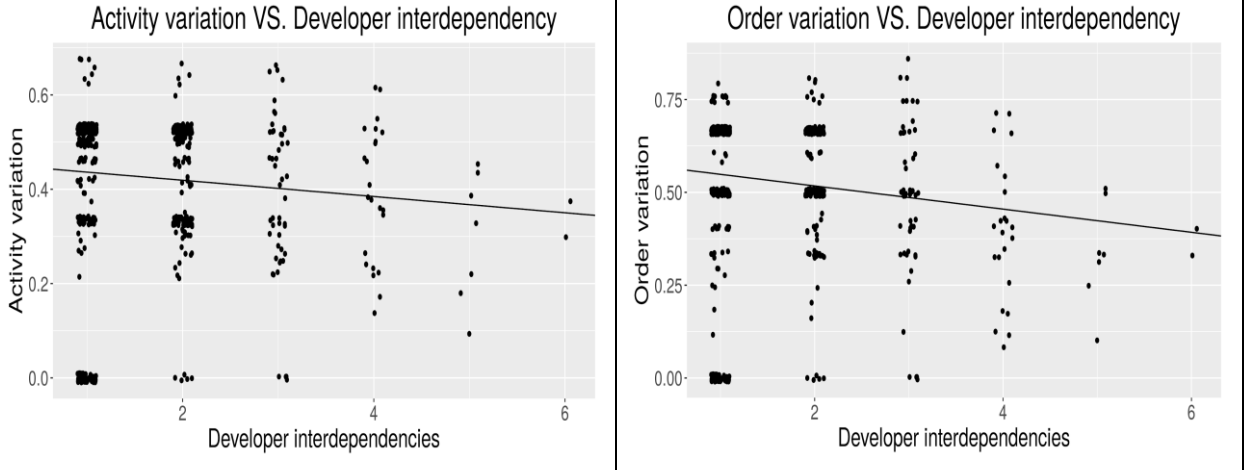


Figure MR. 3: Routine Variation and Developer Interdependencies

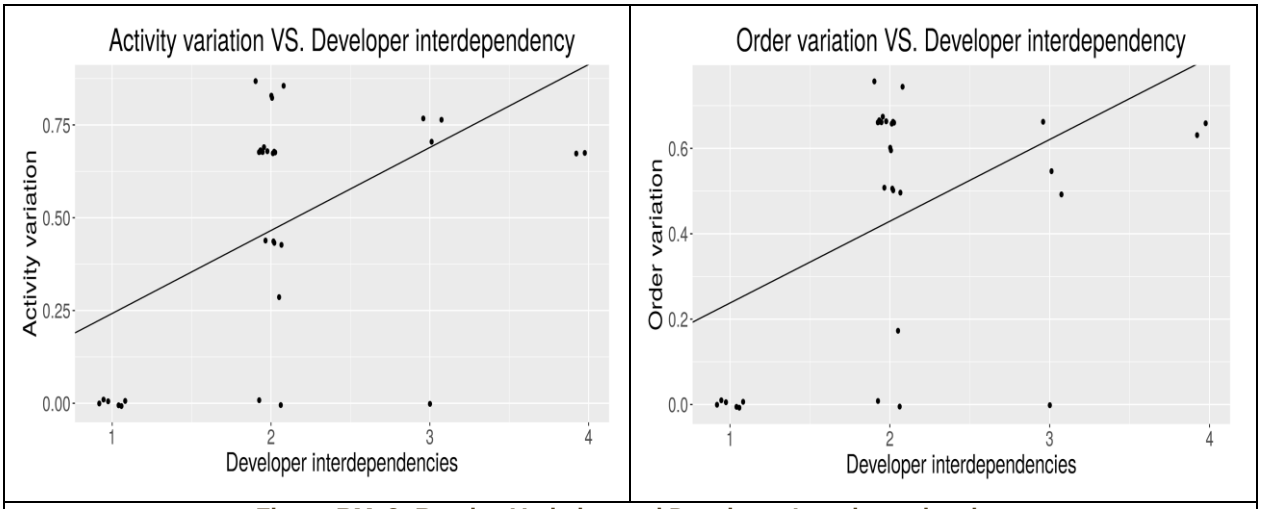


Figure RM. 3: Routine Variation and Developer Interdependencies

Appendix B: Model: activity variation ~ log(development interdependencies) + activity length for JRuby(JR), mruby(MR) and RubyMotion(RM)

Table B1. activity variation ~ log(development interdependencies) + activity length for JRuby(JR), mruby(MR) and RubyMotion(RM)

	Rubinius(Rep.)			Jruby			Mruby			RubyMotion		
	Coeff	p	Adj. R^2	Coeff	p	Adj. R^2	Coeff	p	Adj. R^2	Coeff	p	Adj. R^2
log(Dvpt. intrdp)	0.001	0.843	0.24	-0.0084	0.08	0.0029	-0.0046	0.195	0.0008	-0.1187	0.0116	0.65
No. of evts	0.023	2e-16		0.0012	0.29		0.0016	0.301		0.1052	9.27e-09	

Appendix C: Model: activity variation ~ developer interdependencies + activity length for JRuby(JR), mruby(MR) and RubyMotion(RM)

Table C1. activity variation ~ developer interdependencies + activity length for JRuby(JR), mruby(MR) and RubyMotion(RM)

	Rubinius(Rep.)			Jruby			Mruby			RubyMotion		
	Coeff	p	Adj. R^2	Coeff	p	Adj. R^2	Coeff	p	Adj. R^2	Coeff	p	Adj. R^2
Dvper. intrdp	0.015	0.06	0.069	0.011	0.098	0.10	-0.048	3.1e-9	0.092	0.0165	0.80	0.58
No. of evts	0.006	2e-4		0.010	2e-16		0.019	1.53e-15		0.1038	1.83e-05	

About the Authors

Randy V. Bradley, PhD, CPHIMS, FHIMSS is an Associate Professor of Information Systems and Supply Chain Management and Haslam Family Faculty Research Fellow in the Haslam College of Business at The University of Tennessee. He holds a Ph.D. in Management of Information Technology (IT) and Innovation, an M.S. in Management Information Systems, and a B.S. in Computer Engineering, all from Auburn University. His areas of interest include digital business transformation, supply chain digitalization, and the strategic application of business analytics and IT in the supply chain. His research has appeared or is forthcoming in the Production and Operations Management Journal, IEEE Transactions on Software Engineering, Journal of Business Logistics, Decision Sciences Journal, Journal of Management Information Systems, MIS Quarterly Executive, Information Systems Journal, and Journal of Information Technology, among others.

Audris Mockus, PhD is the Ericsson-Harlan D. Mills Chair Professor of Digital Archeology and Evidence Engineering in the Department of Electrical Engineering and Computer Science at the University of Tennessee. He holds a Ph.D. in Statistics from Carnegie Mellon University and an M.S. and a B.S. in Applied Mathematics from Moscow Institute of Physics and Technology. His primary research interests are digital archaeology, software engineering, and data science in which he studies software developers' culture and behavior through the recovery, documentation, and analysis of digital remains, with a focus on FLOSS projects. Prior to entering academia, he worked at Bell Labs and Avaya Labs.

Yuxing Ma, PhD holds a Ph.D in Computer Science from the University of Tennessee, Knoxville, an M.S. from North China Institute of Computer System Engineering, and a B.S. in Electronic Science & Technology from Beijing Institute of Technology (BIT). His research is focused on open source software engineering. Specifically, leveraging data driven approaches to understand the evolution of open source software ecosystems.

Russell Zaretzki, PhD is an Associate Professor of Business Analytics & Statistics in the Haslam College of Business at The University of Tennessee. He holds an M.S. and a Ph.D. in Statistics from Cornell University, and a B.S. in Physics from University of Michigan. His research focuses on applied statistical modelling, Bayesian statistics, and methods for causal inference.

Bogdan C. Bichescu, PhD is an Associate Professor of Management Science in the Haslam College of Business at The University of Tennessee. He holds a PhD in Operations Management from the University of Cincinnati. His research interests are concentrated in the areas of supply chain modeling and healthcare operations. More specifically, his work examines the role of channel power and subcontracting on supply chain performance, while his recent studies investigate the interplay between information technology and hospital performance.