Do open projects "break the mirror"? : Re-conceptualization of organizational configurations in Open Source Software (OSS) production

Eunyoung Moon and James Howison
School of Information
University of Texas at Austin
Austin, Texas, USA
eymoon@utexas.edu , jhowison@ischool.utexas.edu

ABSTRACT
The mirroring hypothesis predicts that loosely-coupled developers will develop a loosely-coupled software system. However, empirical studies have brought confusing results about the mirroring relationship in open source software (OSS) production: loosely-coupled OSS contributors have developed a tightly-coupled system, deviating from theoretical prediction, but are still successful. This study aims to provide better understanding about "breaking the mirror" in community-based OSS production in which there is no significant corporate participation. We propose it is not the mirroring hypothesis that is broken, but the manner in which we conceptualize and measure organizational configurations in OSS production.

CCS Concepts
• **Human-centered computing→ Collaborative and social computing**

Keywords
Open Collaboration, Open Source Software, Software Design, Organization Design.

1. INTRODUCTION
The relationship between the structure of organization and the structure of the system has been of great interest across many fields including system design and organization design. Theorists have formulated the correspondence between two structure is desirable or ought to exist in order to improve performance, which is known as Conway's law [5]. Cataldo, Herbsleb, & Carley (2008) [3] in their study on socio-technical congruence found that when there was congruence between coordination mechanisms and requirements, software development tasks were more rapidly completed, as measured by the resolution time of modification requests. Colfer & Baldwin (2010) formally define the mirroring hypothesis [4] that predicts the organizational factors—firm and team co-membership, physical co-location, and communication links—will be mirrored in technical dependencies of a system under development. The mirroring relationship between two structures occurs because coordinating product design decisions requires communication among the developers who make those decisions.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than the author(s) must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from Permissions@acm.org.
CHASE'16, May 16 2016, Austin, TX, USA
Copyright is held by the owner/author(s). Publication rights licensed to ACM.
ACM 978-1-4503-4155-4/16/05…$15.00
DOI: http://dx.doi.org/10.1145/2897586.2897593

However, Colfer & Baldwin (2010)'s review on the mirroring hypothesis [4] brought confusing results in open collaboration: loosely-coupled OSS contributors have developed tightly-coupled software systems, but are still successful.

The purpose of this study is to elaborate and re-conceptualize organizational configurations in community-based OSS production. We propose it is not the mirroring hypothesis that is broken, but the manner in which we conceptualize and measure organizational configurations in OSS production. Our preliminary study provides evidence that the mirror was not broken because the mirroring hypothesis held in our close observation of software work, using our re-conceptualization of organizational configuration. We argue that the appearance of a broken mirror stems from conceptualizing organizational structure as if it is static or predetermined.

2. THE RESEARCH OPPORTUNITY
2.1 Current stage of knowledge about the mirroring hypothesis in OSS production
In the context of OSS production, researchers have assumed that OSS contributors share few or no organizational factors because developers are geographically dispersed and rarely meet face-to-face. In this path of reasoning, researchers have identified the organizational form of OSS projects as loosely-coupled, and the organizational form of proprietary software development as tightly-coupled [4;11]. Following the mirroring hypothesis, the emergent architecture of OSS system will have fewer connections between modules than proprietary systems, corresponding to the organizational forms. Without loosely-coupled system design, the argument runs, open collaboration, unable to draw on tightly coupled organization, would break down, resulting in failed projects.

However, a review on the mirroring hypothesis [4] found that many cases of open collaboration clearly challenged the hypothesis, "breaking the mirror". The current stage of knowledge about the mirroring hypothesis is that we have an accumulation of empirical findings that OSS practices deviate from theoretical prediction. However, researchers have not yet developed a compelling theoretical argument to account for unexpected phenomenon. Hence, it still remains a challenging and interesting question: "How are tightly-coupled design decisions or tasks coordinated in the relative absence of shared organizational factors?"

Another study [14] suggests that organizational circumstances—the way the work is done for a particular episode—are "fluid" on a

continuum over time between tightly-coupled and loosely-coupled in OSS production. The authors argue that the fluidity of open development belies efforts to relate organizational structure and product structure because, unlike formal organizations, there may be no particular structure that persists over time. The next section provides greater details of how this study proposes to extend the perspective on organizational circumstances [14] and to test it as an explanation for the ability of successful open projects to appear to "break the mirror".

2.2 Research Questions

Researchers have assumed that organizational forms of community-based OSS production are fixed as loosely-coupled as if they are static and predetermined [e.g., 4;11]. This approach implies that interactions among OSS contributors are shaped by system structure [18], overlooking how interpretation and social interests shape technology production through social interactions among people [16]. Such a persistent assumption stems from the perspective that organizations are "containers" for the work that is done in them [21].

This study seeks to explore "organizational configurations" defined as "any multidimensional constellation of conceptually distinct characteristics that commonly occur together" [12:p.1278]. With a configurational approach, this study proposes to take a dynamic, complex view of how work is organized over time, drawing on the Weick's perspective on organization as organizing [19] and thus changeable (albeit somewhat regularized). Organizational configurations have a nuanced meaning referring to the conditions surrounding the process of organizing. In contrast, structures or forms, which have been used by previous studies, have more connotation of being fixed, and reify them as static, unitary objects.

Taking a fine-grained perspective that distinguishes changes over time in organizing work in an OSS project, we develop and test the hypothesis: what is broken is not the mirroring hypothesis but the manner in which we conceptualize and measure organizing in OSS production.

Our first two research questions address each side of the mirroring hypothesis over time. We first examine the structure of the product over time, and then we examine organizational configurations over time.

RQ1: To what extent does software coupling change over time in an OSS project?

In particular we seek to identify periods in which software coupling changed significantly, rather than mere random drift over time. Given the mirroring hypothesis and the assumption that OSS projects are loosely-coupled organizationally, we specifically identify inter-release periods that lead to the software becoming significantly more tightly-coupled.

Organizational configurations require more detailed, qualitative investigation, accordingly we are able to examine only part of the life of the project. RQ1 enables us to identify periods where significant changes occurred, therefore we ask,

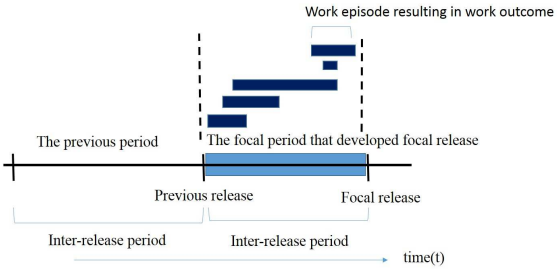
RQ2: What, if any, are the differences in the emergent dimensions of organizational configurations between the period in which software coupling rose (the focal period) and the period just preceding it?

In OSS production, the way that the work is done constitutes the organization [21]. Hence, this study considers that organization is enacted in each episode in which organization is enacted [19;21]. Hence, this study considers that organizational structure, for any time period, of an OSS project is an aggregate of organizational configurations of each work episode (see Figure 1). Here, episodes are defined as "events, processes and practices that occur over time and have a beginning and an end" [1:p.2]. Since our re-conceptualization of organizational configurations is at the episode level, we dive deeper into the data to examine how work was done at the episode level.

RQ3.1: How was work organized for each episode during the focal period?

Finally, to answer the unanswered question from the mirroring hypothesis in open collaboration: "How are tightly-coupled design decisions or tasks coordinated in the relative absence of shared organizational factors?", we seek to investigate whether and to what extent each dimension of organizational configurations is associated with the different organizational process of software work to make a decision about software design [4; 5] at the level of work episode situated in contexts, rather than abstract, deterministic relationships between two structures that transcend settings [15].

RQ3.2: To what extent are each emergent dimension of organizational configurations associated with different organizational process of software work in the work episodes during the focal period?



Note:
We refer to any period in which software coupling rises significantly as a "focal period".

Figure 1. Identification of work episodes that occurred during the specific focal period that developed an OSS system becoming significantly tightly-coupled.

3. EXPLANATORY SEQUENTIAL MIXED METHODS

This study employs explanatory sequential mixed methods. The purpose of the explanatory sequential design is that qualitative data helps explain or build upon initial quantitative results (see Figure 2). In this design, researchers identify specific quantitative findings that need explanation of why these results occurred [6].

We draw on theoretical sampling rather than random sampling, considering the purpose of this study is to discover and account for a phenomenon of interest. Our sampling frame was chosen to maximize the range of information and unexpected insights uncovered by the current stages of our knowledge about the mirroring hypothesis in community-based OSS production.

Theoretical sampling considers the OSS project founded by volunteers. It is significant to note that the term OSS project merely denotes that the projects release the code under open source license. That does not necessarily mean it has been developed by geographically dispersed, sporadically available volunteers. Rather the term OSS encompasses OSS projects initiated by firms in which

a set of developers who share firm co-membership, often work at the same site, and have rich, interpersonal communication through work-related activities within the firm. For this reason, we examine a set of projects making similar applications and calculate the coupling of the software and the size of the contributors; we sample a project that has the largest number of contributors despite the highest software coupling.

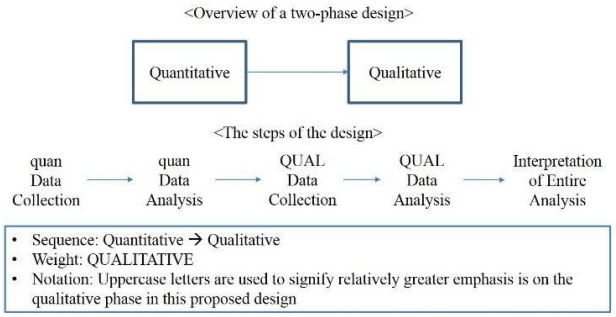


Figure 2. Explanatory sequential design [Adapted from [7] Figure 4.3]

3.1 Phase 1

The goal of phase 1 is to identify the relevant phenomenon, the specific focal periods in which a system became significantly tightly-coupled. To do so, we collect and analyze measurement of software coupling over all releases within sampled OSS projects. To obtain software coupling, we produce a Design Structured Matrix (DSM) with Scitools[1] at the source file level. Then, we wrote a program in Numpy to implement the algorithm, [11;13] called "propagation cost". Then, we employ statistical tests on the series of coupling measures: 1) cluster analysis and 2) a subsequent test to identify the specific focal periods that developed a system becoming significantly tightly-coupled (**RQ1**). If underlying distribution of data that emerge from cluster analysis satisfies the assumption of parametric test, we perform t-test or ANOVA. Otherwise, we use non-parametric test such as Mann-Whitney U test or Kruskal-Wallis H test (Table 1).

Table 1. A subsequent test to be performed, depending on underlying distribution of cluster that emerges from the Cluster Analysis

Number of Cluster	Parametric test	Non-parametric test
Two	t-test	Mann-Whitney
More than two	One-way ANOVA	Kruskal-Wallis

3.2 Phase 2

3.2.1 Release level analysis

Once the specific focal periods are identified, our attention turns to gaining a broad understanding of organizational configurations between the inter-release periods: 1) the previous period right before software coupling became significantly tightly-coupled, and 2) the focal period that resulted in a dramatic increase in software coupling (see Figure 1). We focus on characterizing the organizational configurations to compare between the successive periods (**RQ2**). To do so, we collect multiple sources of public archival data to familiarize ourselves with the work done in the period. We then crafted an interview protocol to explore how work was done. Taking a semi-structured, artifact-based approach, we tailored our interview protocol for each interviewee, and interview OSS contributors who made most of commits during the previous period, those who made most of commits during the focal period, and those who made commits for both periods, for relative comparison of organizational configurations at the release level.

3.2.2 Work episode level analysis

We identify work episodes resulting in work outcomes that occurred during the focal period. The work episodes are identified from multiple sources of qualitative data including public archival records (e.g., the Release Notes, the README file, and Pull Requests), patch messages sent to the project mailing lists, commit logs, and semi-structured artifact-based interviews. Here, work outcomes are defined as *changes made to the software project's shared source code repository for a particular work episode*, building on Howison's (2009) [9] definition of task outcome—"the actual change to the group's shared outputs which occurs as a result of the work directed towards it" (p.73). We define a work episode resulting in work outcomes as *the process to make a set of commit(s) such as a patch (a Pull Request) or a direct check-in of the code, which is applied to the project source code repository*.

We define a beginning of a work episode that triggered creating a patch (or a Pull Request), for instance, user's feature requests or bug reports. However, not all patches are initiated by users' feature requests or bug reports. In that case, submitting a patch (or a Pull Request), which was initiated by the author of the patch, triggers a work episode. The process of working on the submitted patch (or the Pull Request) by reviewing it and asking for tweaks, captures the middle of a work episode. Finally, applying the patch (or the Pull Request) to the source code repository is an ending of a work episode. In other cases, OSS developers may not send a patch message or a pull request but directly check-in the code. In those cases, we will analyze the commit logs to investigate what led them to write the code (see Figure 3).

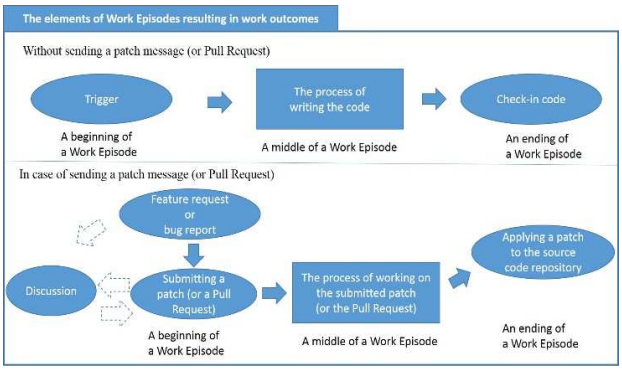


Figure 3. The elements of Work episodes resulting in work outcomes

To create work episodes resulting work outcomes, we extract all commits made during the focal period. Then, we group a set of commit(s) into a work episode resulting in a work outcome, using a spreadsheet program. Once we exhaustively create the work episodes, we seek to explore the emergent dimensions of organizational configurations in the work episodes during the focal

[1] The tool is distributed by Scientific Toolworks, Inc. www.scitools.com

period(s) that developed the OSS system becoming significantly tightly-coupled (**RQ3.1**) (see Figure 1). In doing so, we start with three dimensions of organizational configurations from literature: 1) work co-membership outside the OSS project, 2) physical and temporal co-location, and 3) rich, interpersonal communication [4], allowing other dimensions to emerge from data. Then, we calibrate each dimension of organizational configurations, quantitizing qualitative data [17]. We assign the value in the interval between 0 and 1 for each dimension of organizational configurations in the work episodes. For instance, we compute the ratio of OSS contributors involved in the episode that shared membership in an organization (e.g., a firm). For instance, if three contributors were involved in the work episode 1, and two contributors had work co-membership outside the OSS projects, working for the same company, then, we will assign the value of 0.67. The continuous value 0.67 denotes work co-membership partially shared among those who involved in the work episode 1. Then, we convert it categorical value—"more presence" in work co-membership. In this way, we will calibrate physical and temporal co-location to compute the ratio of those who were co-located in the work episodes, and then, convert it into categorical data. For other dimensions of organizational configurations such as division of labor, and rich, interpersonal communication, we will assign the value of 0 or 1 to denote absence or presence. For instance, based on archival data records which work outcomes were produced by whom, we will assign the value 0 or 1 in the work episodes. The example of categorical values to denote the degree of absence (or presence) of organizational configuration dimensions is shown in Table 2.

Table 2. Converting the continuous value into the categorical value for each emergent organizational configuration dimensions in the work episodes

The continuous value	Categorical value
1.0	Full presence
0.80	Mostly presence
0.60	More presence
0.40	More absence
0.20	Mostly absence
0.00	Full absence

Next, we seek to identify and analyze the patterns between the dimensions of organizational configurations and the organizational process of software work to understand how work was actually done (**RQ3.2**). To do so, we code the organizational process of software work in the work episodes "tightly-coupled", if it was impossible to make a decision about software design without ongoing interaction with others. We code it "loosely-coupled", if there was simple acknowledgement to merge the submitted patch or notification that commits were pushed without having to affect other's work or ongoing interaction. However, we were not able to classify all work episodes as either tightly-coupled or loosely-coupled due to the nature of the work. For instance, updating copyright or authorship does not need making a decision about software design. In this case, we code it as "Neither". Then, we perform the Chi-square test to see whether there is any significant relationship between each dimension of organizational configurations and different organizational process of software work with respect to whether it required to make decisions about software design or not. If it is significant, we report to what extent two variables are associated with each other with Phi coefficient. Given that these variables are measured categorical data, the chi-square test is well-suited to investigate if there is a relationship between two categorical variables—each dimension of organizational configurations and organizational process of software work.

3.3 Interpretation phase

In this stage of analysis, we merge the findings from phase 1 and phase 2. We look for similarity and differences of organizational configurations in which developed a system becoming significantly tightly-coupled across different projects [8]. This comparison will be based on a combination of the quantitative statistical results, the qualitative findings about organizational configurations, and the association between organizational configurations and the organizational process of software work.

4. PILOT STUDY: GNU grep

We present our findings from our pilot study of GNU grep. GNU grep was identified as the project that has the largest number of contributors despite the highest level of software coupling in command-line searching utility—GNU findutils[2], the silver searcher[3], regexxer[4] (See Figure 4). The GNU grep project is not a large project in terms of Lines of Code (LoC), however, it is reasonable to study, given that a vast majority of OSS projects are either small or medium, and only a minor fraction of projects are large [2].

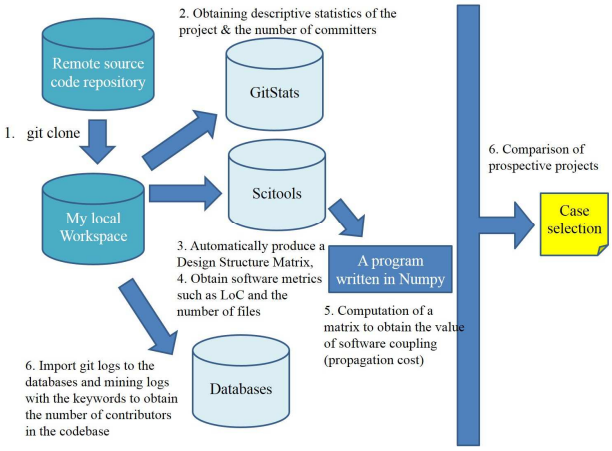


Figure 4. The process to identify the project that has the large number of contributors despite the highest level of software coupling

4.1 Phase 1

All releases of GNU grep available on git and the main GNU ftp server were downloaded. We downloaded a total of 28 releases over 20 years. Using a Scitool, a DSM for each release was produced at the source file level. Then, each matrix was computed to obtain the

[2] http://savannah.gnu.org/git/?group=findutils

[3] https://github.com/ggreer/the_silver_searcher

[4] http://regexxer.sourceforge.net/

value of software coupling with a program written in Numpy that implement the algorithm of propagation cost. All values of software coupling were entered into SPSS program, and we obtained a graph that tracks the evolution of grep's software coupling over all releases (see Figure 5).

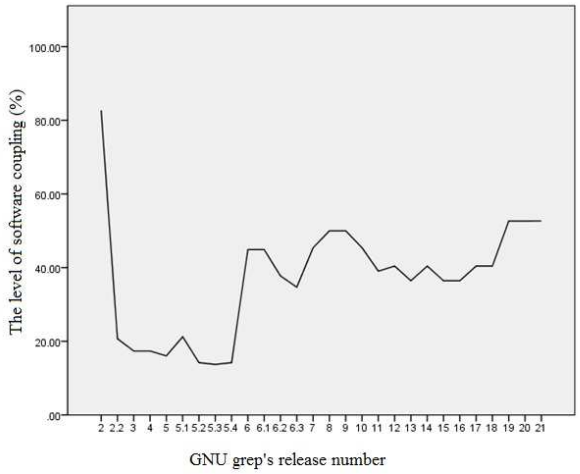


Figure 5. The evolution of grep's software coupling over releases. [Note: on the X-axis the big number 2 was omitted to increase readability]

A hierarchical cluster analysis using Ward's method and squared Euclidean distance produced three clusters. Because the underlying distribution of three clusters did not satisfy the assumption of ANOVA, we performed a Kruskal-Wallis H test to see whether there are statistically significant differences among clusters on the level of software coupling. It resulted in the inter-release period between grep 2.5.4 and grep 2.6 is the focal period developing a system significantly tightly-coupled. Closer inspection on software metrics reveals that there was a dramatic decrease in the size, and large structural changes, removing old, built-in libraries and importing GNU libraries (see Table 3).

Table 3. Comparison of Software metrics between the previous release (grep 2.5.4) and the focal release (grep 2.6)

	Previous	Focal
Software coupling (%)	14.21	44.90
Lines of Code	31,313	8,831
Number of source files	86	22
Number of functions	261	123
Inactive Lines	7,096	1,431
Preprocessor Lines	5,177	560

4.2 Phase 2

4.2.1 Release level analysis

To explore the emergent dimensions of organizational configurations, we first sought to gain a broad understanding of two inter-release periods: the previous period (grep 2.5.4) and the focal period (grep 2.6). Based on the analysis of qualitative data from public archival data including messages exchanged via bug-grep mailing list[5] during two inter-release periods, we crafted the generic interview protocol that asks about the history of participation in the project, specified partitioning in the codebase and the division of labor, and organizational configuration dimensions that organized software work to apply a set of commit(s) during the inter-release periods. Then, we employed a semi-structured artifact-based approach to interview a grep 2.5.4 maintainer who made most of commits (28 out of 42 commits) during the previous period and one of two grep 2.6 maintainer who made most of commits (207 out of 218 commits) during the focal period. For each interviewee, we tailored the generic interview protocol to help each interviewee achieve fuller development of information [20]. In conducing the interview, we showed a list of work outcomes and particular commits made by the grep developers to the interviewees. The interview with grep 2.5.4 maintainer was conducted via Skype and then, transcribed verbatim. The interview with one of two grep 2.6 maintainers was conducted via email.

First, to see whether there was work co-membership outside the project, we looked for the use of affiliation email address when those maintainers used to post messages or make commits. We found that one grep 2.6 maintainer used affiliation email address and our interview revealed that grep 2.5.4 maintainer didn't have work co-membership with other grep contributor who made 14 out of 42 commits. In case of grep 2.6 maintainers, they worked for the same company. Second, as to the physical and temporal co-location, there was no public archival records that indicate grep developers had co-located events. The interview confirmed that there were no co-located events, and that grep 2.6 maintainers were not co-located at the same site to work on the project. Third, the interviews revealed that two grep 2.6 maintainers had rich, interpersonal communication during the focal period. The interview revealed that two grep 2.6 maintainers talked a lot via private email outside the project, as well as bug-grep mailing list during the focal period to discuss about grep work. Further, we noted that there was an explicit division of labor and a broad roadmap in advance during the focal period, based on the analysis of email messages exchanged via bug-grep. In contrast, the interview with grep 2.5.4 maintainer reveals that there was no roadmap during the previous period, and grep contributors basically went through the patches that were submitted to the patch manager system.

Table 4. Relative comparison of organizational configuration dimensions between two inter-release periods

	Previous	Focal
Work co-membership	No	Yes
Co-location	No	No
Rich, interpersonal communication	No	Yes
Division of labor	No	Yes
	No roadmap: Going through patches, as people submit, The project mailing list as a primary	Worked with a roadmap, Explicit division of labor,

[5] http://lists.gnu.org/archive/html/bug-grep/

	means of communication	Lots of use of private email

In conclusion, qualitative analysis of archival records and interviews reveals that there was a change in organizational configurations from the previous period to the focal period. Table 4 is a summary of data analysis.

Nonetheless, it's important to note that the interviews in this stage explored gaining a broad understanding of organizational configurations at the level of release. Since our re-conceptualization of organizational configurations is at the episode level, we sought to examine how work was done in the work episodes during the focal period.

4.2.2 Work Episode level analysis

We extracted 218 commits made during the focal period (grep 2.6). Then, we classified those commits into work outcomes, which we compiled from the analysis of qualitative data including patch message sent to bug-grep mailing list, bug tracker, patch manager, and the Release Notes. In case that commits were not sent to bug-grep or patch tracker system but directly applied to the source code repository, we created new work outcomes and created a work episode that made such commits. This process resulted in a total of 36 work episodes.

For each work episode, we recorded the dimensions of organizational configurations: work co-membership, physical and temporal co-location, rich, interpersonal communication, and division of labor. If those who involved in the work episodes had work-related activities outside the project to produce work outcomes or shared perspectives or resources (e.g., test cases) that come from work co-membership, then we recorded there was work co-membership for that specific work episode. Also, we computed the ratio of work co-membership shared among those who involved in the work episode. Then, we recorded whether there was discussion about the submitted patch (or the Pull Request) before it was finally applied to the source code repository, or whether there was a simple notification that the particular patch was just applied to the source code repository. For instance, we recorded if there was discussion about certain dependencies of their own work, ongoing clarification from others, reviews before proceeding with their work, or they mentioned that they had talked about the work outside the project mailing list (e.g., private mails or off-list). Those work episodes were recorded as the ones that had rich, interpersonal communication, and we assigned the value of 1. In contrast, the work episodes that have the patches sent to the project mailing list for simple notification are classified as the ones that did not have rich, interpersonal communication, and we assigned the value of 0. The work episodes for which patches were applied to the source code repository without even simple notification were also classified as not having rich, interpersonal communication, and we assigned the value of 0.

In this section, we present an illustrative work episode in which two maintainers worked in a tightly-coupled way. 9 out of 36 work episodes (56 out of 218 commits) were related to switching from built-in libraries to GNU libraries, which is termed "Gnulibification". Gnulibification required two grep maintainers to manually sync with gnulib modules to take advantage of using updated gnulib modules as of the focal period. For instance, the work episode resulting in "Gnulibification: The sync with GNU awk" illustrates the process to sync grep's source file (dfa.c) to one of GNU libraries, GNU awk (GAWK). In the process of adaptation to GAWK, two maintainers exchanged messages via bug-grep mailing list, and shared a test case that came from the bug tracker system of the company that both developers worked for. In this work episode, the patch to adjust API from GAWK required two grep 2.6 maintainers to further discuss it. This work episode illustrates tightly-coupled software work that led two grep maintainers to share and discuss design-relevant information and solutions.

In short, the illustrative work episode suggests that two maintainers relied on work co-membership outside the project that enabled them to share company's test case easily, direct communication, and private emails.

Next, as we stated in section 3.2, we coded the organizational process of software work in the work episodes with respect to whether it required making a decision about software design with other developers (tightly-coupled software work), or there was simple acknowledgement or notification (loosely-coupled software work). This stage of analysis resulted in 16 work episodes were tightly-coupled software work, 13 work episodes were loosely-coupled software work, and 7 work episodes were not either tightly-coupled or loosely-coupled due to the nature of the work (e.g., updating copyright, updating THANKS) (see Table 5).

Table 5. Descriptive statistics of organizational process of software work in the work episodes

Organizational process of software work	Frequency	Percent
Tightly-coupled	16	44.4
Loosely-coupled	13	36.1
Neither	7	19.4
Total	36	100

We performed a Chi-square test to examine the association between each dimension of organizational configurations and the organizational process of software work in the work episodes. We do not sum up the values of organizational configuration dimensions but keep them distinct in the work episodes because this study views organization is enacted in each work episode [19].

The test reveals that there appears to be an association between work co-membership and organizational process of software work (χ^2=12.857, $p<0.05$), and it was a moderately strong relation (Phi coefficient = 0.598). Work co-membership among those who involved in the work episodes was more likely to present in tightly-coupled software work. The relationship between rich, interpersonal communication and organizational process of software work was significant, and it was a moderately strong relation (χ^2=16.812, $p<0.0005$, Phi coefficient= 0.683). Rich, interpersonal communication among those who involved in the work episodes was more likely to present in tightly-coupled software work. The division of labor was not significantly associated with the organizational process of software work. In case of physical and temporal co-location, statistics was not computed because none of the work episodes were distributed on this dimension.

4.3 Interpretation phase

A dramatic, statistically significant increase in the level of software coupling during the focal period might be considered theoretical deviation "breaking the mirror", if researchers assume the organizational form of OSS is fixed and loosely-coupled. However, this period saw more work co-membership outside the project and rich, interpersonal communication via mailing list, and private

email. In addition, two grep 2.6 maintainers shifted towards traditional coordination mechanisms such as working with a roadmap, explicit division of labor, and relying on firm co-membership. This is contrasted with the previous assumption that in OSS projects virtually all interchange among contributors takes place in public, and is transparent [4]. The previous period building grep 2.5.4 is described absence of organizational configuration dimensions, no roadmap in advance, communication only via the public mailing list. Activity continued but focused on tasks that could be accommodated by existing organizational configurations, such as incremental bug-fixes, which appears to the superposition of individual work [10]. That is to say, the coupling of organizational configurations at the level of release ecologically moved from loosely-coupled to more tightly-coupled, corresponding to changes in the software coupling. At the level of work episodes we presented an illustrative work episode in which grep maintainers worked in a tightly-coupled way. A statistical test at the level of work episodes revealed that the emergent dimension of organizational configurations—work co-membership outside the project and rich, interpersonal communication—were moderately associated with tightly-coupled software work which required to make a decision about software design among those who involved in the work episodes.

5. CONCLUSION

This study proposes to take a more fine-grained perspective on organizational configurations of OSS production, whereas other researchers have assumed a pre-determined, fixed organizational form. Our proposed re-conceptualization of organizational configurations should help elaborate organizational forms of OSS production in a more nuanced way. At least it enables us to better state questions of why the mirror might seem to be broken in OSS production. Ultimately, it enables us to understand how OSS practice deviates from expected patterns but why the projects are still successful.

6. REFERENCES

[1] Annabi, H., Crowston, K., & Heckman, R. (2008). Depicting What Really Matters: Using Episodes to Study Latent Phenomenon. *ICIS 2008 Proceedings*. Retrieved from http://aisel.aisnet.org/icis2008/183

[2] Capiluppi, A., Lago, P., & Morisio, M. (2003). Characteristics of open source projects. In *Seventh European Conference on Software Maintenance and Reengineering, 2003. Proceedings* (pp. 317–327). http://doi.org/10.1109/CSMR.2003.1192440

[3] 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. In *Proceedings of the Second ACM-IEEE International Symposium on Empirical Software Engineering and Measurement* (pp. 2–11). New York, NY, USA: ACM. http://doi.org/10.1145/1414004.1414008

[4] Colfer, L., & Baldwin, C. Y. (2010, February 18). The Mirroring Hypothesis: Theory, Evidence and Exceptions — HBS Working Knowledge. Retrieved September 27, 2014, from http://hbswk.hbs.edu/item/6361.html

[5] Conway, M. E. (1968). How do committees invent. *Datamation*, *14*(4), 28-31.

[6] Creswell, J., Clark, V., Gutmann, M., & Hanson, W. (2003). *Advanced mixed methods research designs*. SAGE.

[7] Creswell, J., & Clark, V. L. P. (2007). *Designing and Conducting Mixed Methods Research*. SAGE.

[8] Eisenhardt, K. M. (1989). Building Theories from Case Study Research. *Academy of Management Review*, *14*(4), 532–550. http://doi.org/10.5465/AMR.1989.4308385

[9] Howison, J. (2009). *Alone together: A socio-technical theory of motivation, coordination and collaboration technologies in organizing for free and open source software development* (Ph.D.). Syracuse University, United States -- New York.

[10] Howison, J., & Crowston, K. (2014). Collaboration Through Open Superposition: A Theory of the Open Source Way. *MIS Q.*, *38*(1), 29–50.

[11] MacCormack, A., Baldwin, C., & Rusnak, J. (2012). Exploring the duality between product and organizational architectures: A test of the "mirroring" hypothesis. *Research Policy*, *41*(8), 1309–1324. http://doi.org/10.1016/j.respol.2012.04.011

[12] Meyer, A., Tsui, A., & Hinings, C. R. (1993). Configurational Approaches to Organizational Analysis. *Academy of Management Journal*, *36*(6), 1175–1195.

[13] Milev, R., Muegge, S., & Weiss, M. (2009). Design Evolution of an Open Source Project Using an Improved Modularity Metric. In C. Boldyreff, K. Crowston, B. Lundell, & A. I. Wasserman (Eds.), *Open Source Ecosystems: Diverse Communities Interacting* (pp. 20–33). Springer Berlin Heidelberg.

[14] Moon, E., & Howison, J. (2014). Modularity and Organizational Dynamics in Open Source Software (OSS) production. *AMCIS 2014 Proceedings*. Retrieved from http://aisel.aisnet.org/amcis2014/SocioTechnicalIssues/GeneralPresentations/9

[15] Orlikowski, W. J. (1992). The Duality of Technology: Rethinking the Concept of Technology in Organizations. *Organization Science*, *3*(3), 398–427. http://doi.org/10.1287/orsc.3.3.398

[16] Orlikowski, W. J. (2000). Using Technology and Constituting Structures: A Practice Lens for Studying Technology in Organizations. *Organization Science*, *11*(4), 404–428.

[17] Sandelowski, M., Voils, C. I., & Knafl, G. (2009). On Quantitizing. *Journal of Mixed Methods Research*, *3*(3), 208–222. http://doi.org/10.1177/1558689809334210

[18] Scranton, P. (1995). Determinism and Indeterminacy in the History of Technology. *Technology and Culture*, *36*(2), S31–S53. http://doi.org/10.2307/3106689

[19] Weick, K. E., Sutcliffe, K. M., & Obstfeld, D. (2005). Organizing and the Process of Sensemaking. *Organization Science*, *16*(4), 409–421. http://doi.org/10.1287/orsc.1050.0133

[20] Weiss, R. S. (1995). *Learning From Strangers: The Art and Method of Qualitative Interview Studies*. Simon and Schuster.

[21] Winter, S., Berente, N., Howison, J., & Butler, B. (2014). Beyond the organizational "container": Conceptualizing 21st century sociotechnical work. *Information and Organization*, *24*(4), 250–269. http://doi.org/10.1016/j.infoandorg.2014.10.003