How Developers' Collaborations Identified from Different Sources Tell us About Code Changes

Sebastiano Panichella[1], Gabriele Bavota[1], Massimiliano Di Penta[1], Gerardo Canfora[1], Giuliano Antoniol[2]
[1]Dept. of Engineering, University of Sannio, Italy, [2] École Polytechnique de Montréal, Canada

***Abstract*—Written communications recorded through channels such as mailing lists or issue trackers, but also code co-changes, have been used to identify emerging collaborations in software projects. Also, such data has been used to identify the relation between developers' roles in communication networks and source code changes, or to identify mentors aiding newcomers to evolve the software project. However, results of such analyses may be different depending on the communication channel being mined. This paper investigates how collaboration links vary and complement each other when they are identified through data from three different kinds of communication channels, i.e., mailing lists, issue trackers, and IRC chat logs. Also, the study investigates how such links overlap with links mined from code changes, and how the use of different sources would influence (i) the identification of project mentors, and (ii) the presence of a correlation between the social role of a developer and her changes. Results of a study conducted on seven open source projects indicate that the overlap of communication links between the various sources is relatively low, and that the application of networks obtained from different sources may lead to different results.**

***Keywords*—Developers, Developer Social Network, Empirical Study**

I. Introduction

The communication among projects' members plays a paramount role in any successful software project. Indeed, team coordination and communication has always been the crux of people involved in software project management [1]. Notwithstanding the nature of a project (*i.e.*, open source versus industrial/closed source), its domain, or size, the involved people need to exchange information effectively, minimizing the communication overhead and making sure they are up to date with the project status.

In everybody's experience, different communication channels play different, sometimes complementary sometimes alternative, roles: news can be gathered from the radio, by reading a newspaper, watching a TV broadcast or surfing blogs. Each channel has its pros and cons: TV/radio tend to be timely; Internet in addition has less control; newspapers could provide a deeper and focused treatment of some topics. Besides that, which communication channel is preferred is a mere personal choice influenced by various factors, such as the information need, the age, the culture or the life style. Much in the same way, people contributing to a project may prefer a particular communication channel. For example, general discussions about a project's perspective, software design, or future development strategies may happen in mailing lists, whereas discussions related to specific features or to the resolution of bugs occur on issue trackers. Another factor is the size, structure and general organization of the project. For example, some projects tend to have in the past most of the discussion over mailing lists, and only in recent years they tend to use issue trackers much more. Finally, in industrial projects part of the discussion occurs through face-to-face or phone meetings [2].

In recent and past years, (written) communication has been analyzed by several authors for different purposes and exploited to support software evolution tasks. For example, Bird *et al.* [3] and Hong *et al.* [4] studied to what extent emerging teams identified from email and issue tracker communication reflect the latent structure of software projects. Bird *et al.* [5] found a correlation between social network metrics and change activities. Finally, Bettenburg *et al.* [6] and Kumar *et al.* [7] studied how social network metrics could be used for bug prediction purposes. Canfora *et al.* [8] used data from mailing lists and issue trackers to recommend mentors.

The studies mentioned above have analyzed projects' communication by observing one or two sources of communication. The conjecture we want to investigate is that, *different communication channels would provide different views of developers' interaction. As a consequence, the use of such information in recommender systems could produce different results*.

To this aim, we analyze written communication between developers (*i.e.*, people changing the code) recorded through mailing lists, issue trackers, IRC chat logs, and code co-changes. The overarching goal is to provide evidence that by analyzing a single communication channel one may obtain a misleading portrait of people interaction, and that in general different combinations of the sources may provide different views of the project's interaction.

By analyzing the communication occurring in seven open source projects we show that (i) not all developers use all communication sources; (ii) people interacting using a given channel may or may not communicate through other channels; (iii) the identification of key project roles—such as developers with a high communication degree or mentors [8]—leads to different results if done over different communication channels; (iv) a study performed in the literature [5] would have achieved different findings when looking into different communication channels.

Paper structure. Section II presents the details of the empirical study design, selected system, approach adopted to collect and analyze data. Section III reports empirical findings and is followed by Section IV where we discuss the threats to validity. After a discussion of related work in Section V, Section VI concludes the paper and outlines directions for future work.

TABLE I. CHARACTERISTICS OF THE ANALYZED PROJECTS.

Project	URL	Year Started	Observed Period	Size (KNLOC)	#Commits	#Comments in issue tracker	#Emails in mailing list	#Messages in IRC chat
Apache HTTPD	http://httpd.apache.org/	1996	June 2011-June 2013	2,021-2,240	4,315	1,659	5,487	640,471
Apache CXF	http://cxf.apache.org	2005	June 2011-June 2013	593–771	4,911	6,016	3,049	305,802
Hibernate	http://hibernate.org	2003	June 2011-June 2013	984–1,096	1,805	992	2,423	84,218
Infinispan	http://infinispan.org	2009	June 2011-June 2013	146–286	2,482	9,305	3,886	893,780
Apache Lucene	http://lucene.apache.org	2000	June 2011-June 2013	198–437	2,957	68,055	10,821	104,901
Samba	http://www.samba.org	1996	June 2010-June 2012	1,278–1426	11,151	9,132	9,979	17,591
Weld	http://weld.cdi-spec.org	2008	June 2011-June 2013	108–139	1,225	1,996	2,423	98,044
Total	–	–	–	–	28,846	97,155	38,068	2,144,807

II. EMPIRICAL STUDY DESIGN

The *goal* of the study is to analyze developers' collaborations mined from different sources of information, with the *purpose* of understanding their commonalities and differences. The *perspective* is of researchers interested in studying to what extent using different sources could produce a different view of how developers interact in a project during its evolution. When such a view is used in the context of empirical studies—*e.g.*, to verify if the number of code-changes performed by developers is related to their activity in the social network—or to build different kinds of recommenders—*e.g.*, to suggest mentors—this could produce different results.

The *context* of the study consists of data from seven open source projects, whose characteristics are summarized in Table I. In particular, Table I reports: an URL linking to the project website, the date when the project started, the observed time period, the code size in terms of non-commented KLOC (KNLOC), and the size of data from the four sources of information. HTTPD is an open-source HTTP server for modern operating systems. CXF is a framework providing APIs for web service development while HIBERNATE is an object-relational mapping library for Java. INFINISPAN is a data grid platform written in Java and designed to be highly scalable. LUCENE is a Java-based indexing and search technology. SAMBA is a re-implementation of the SMB/CIFS networking protocol mostly written in C. Finally, WELD is an implementation of the Contexts and Dependency Injection for Java EE. On the one hand, we have chosen such projects to ensure enough diversity in terms of size (of the code base, of the developers' population and of the exchanged messages). On the other hand, we looked for projects having the availability of data from the four investigated sources—versioning systems, issue trackers, mailing lists, and IRC logs—for a period of at least two years; we deemed two years long enough to observe collaborations.

A. Research Questions

In the context of the study, we formulated the following research questions:

- **RQ$_1$:** *To what extent do developers discuss through the different communication channels?* The conjecture is that some developers may use a limited set of the available communication channels. For instance, it may happen that only a small "core" team actually discusses through IRC, while many more may discuss over the issue trackers.
- **RQ$_2$:** *How do the inferred links between developers overlap when using different sources of information?* This research question investigates whether different sources of information provide a different view of the project social network or, in other words, of the project's members interactions.
- **RQ$_3$:** *How do social network metrics change when using different sources, and how would this impact on using such information to build recommenders?* In this research question we analyze to what extent (i) recommenders aimed at identifying people having some particular role in the project—such as developers having a high degree in the communication or mentors [8]—produce different results when using different sources of information, and (ii) how does the correlation between social network metrics and developers' activity (i.e., how many commits they perform) change when using different sources. To this aim, we replicate, using different sources, an empirical study previously performed by Bird *et al.* [5].

B. Data Extraction Process

This section describes the data extraction process that we follow with the aim of collecting the data needed to perform our study.

1) Downloading the Four Sources of Information: Commits checked in by developers are collected by mining the change log of the **versioning system** hosting the seven subject projects. Note that the versioning system adopted for the analyzed systems, *i.e.*, Git, provides explicit information for authors, other than just for committers, although in many cases authors and committers match. In particular, for each commit we stored: (i) the project's member performing it, (ii) the involved files, and (ii) the commit date. In total, 28,846 commits have been downloaded.

Issue trackers are mined with the aim of extracting developers' discussions carried out on this communication channel. In particular, for each system we download all issues created in the analyzed time period (see Table I) regardless their type (*e.g.*, bug, new feature, etc.) and status (*e.g.*, closed, open, etc.). To perform such a task we built two crawlers for the *Bugzilla* issue tracker (used by SAMBA) and *Jira* (used by the other six projects). For each issue, both crawlers extract (i) the name of the project member posting the issue, (ii) the issue title, (iii) the issue description, (iv) the posting date, and (v) the comments left by project members to the issue, storing for each of them the name, the date, and the message. In total, we collected 5,790 issues comprising 97,155 comments.

Development mailing lists are downloaded from the Web, either by downloading available archives (HTTPD, SAMBA, HIBERNATE, WELD, INFINISPAN), or by crawling Web-based mailing list (LUCENE and CXF). Then, emails are parsed to extract, for each message: (i) the message ID, (ii) the project's member sending the email (*i.e.*, the *from* email field), (iii) the project member(s) to which the email was sent (*i.e.*, the *to* email field), (iv) the ID of the message being replied, (v) the email subject, (vi) the email timestamp, and (vii) the message body. In total, 38,068 emails have been collected.

IRC chats are mined from the Web. In particular, for each discussion thread (reported in a separate page of the chat log) we store: (i) the (nick)name of developers taking part in the discussion, (ii) the thread date, and (iii) the messages exchanged in the thread. In total, 2,144,807 messages have been downloaded.

2) Unifying Project Contributors' Names: We use an approach similar to the one used by Bird *et al.* [5] and used in our previous works [8], [9]. The approach is composed of the following steps:

1) **Normalization:** names are converted into lower cases, and special characters, including dots ".", are removed.
2) **Ignore middle names**, *e.g.*, *john p Smith* corresponds to *john smith* unless this leads to an ambiguity.
3) **First name referred with initials only**, *e.g.*, *john smith* corresponds to *j smith*, unless this generates an ambiguity.
4) **Last name only**, *e.g.*, *john smith* corresponds to *smith* unless this generates an ambiguity.
5) **Initials only**, *e.g.*, *j s* correspond to *john smith*, unless this generates an ambiguity.
6) **User ID-like name:** IDs, often used in versioning composed by concatenating first and last names (or their initials). For example, *john f. smith* could be referred as *johnsmith*, *jsmith*, or *jfsmith*. Again, we check if the same ID can be obtained from multiple persons' names.

To deal with cases where email addresses are used in the project's members' communication, we use a set of heuristics, mostly derived from the above ones, to associate emails to names:

1) **Extract name:** first, we extract the name from the email address, *i.e.*, anything preceding the "@" and split it into terms considering special characters as separators.
2) **Map email address to a name:** we try to map the name extracted from the email to full names occurred in other emails. For example, *jsmith@google.com* is mapped to *John Smith*, even if he was previously associated with a completely different email address.
3) **Map multiple email addresses of the same person:** we map multiple email addresses applying—on the name extracted from the email address—the same heuristics defined for names.

Overall, it is worthwhile to point out that the adopted approach for unifying names and email is a conservative one, *i.e.*, it performs an unification only when there are no multiple (ambiguous) possibilities of unification for the same name. Since we have no guarantee that the aforementioned approach is 100% accurate and complete, we integrated it with a manual analysis performed by two of the authors, aimed at verifying the existing mappings and adding missing ones. Such an analysis lasted four working days, and helped to fix less than 5% of wrong mappings and to add about 20% of missing ones.

3) Extracting Developers' Links: Once unified the names, **we restrict our attention to commit authors'** only. This is because we want to focus our attention to discussions occurring between people involved in code changes only, rather than other people participating to the discussions.

Given two project's members, M_i and M_j, we identify a link $M_i \leftrightarrow M_j$ between them in the four sources of information by applying the following heuristics:

- **Versioning system:** M_i and M_j modify the same file during a specific time interval, fixed in this work to six months. Bear in mind this is not really communication, however it has been used in some past studies [10], [11]. We considered the six months period as not so short (otherwise it would be unlikely to find links) nor so long that the two contributions were completely detached.
- **Issue tracker:** M_i and M_j left a comment to the same issue, M_i left a comment to an issue created by M_j (or *vice versa*).
- **Mailing list:** Both M_i and M_j sent emails / replied to the same email thread [5]. Emails belonging to the same thread have been identified by looking at the message ID of the email itself (for the email opening a thread) and the message ID of the email being replied.
- **IRC chat:** M_i and M_j take part in the same discussion thread.

C. Analysis Method

This subsection describes the analyses and statistical procedures used to address the three research questions formulated in Section II-A.

To address $\mathbf{RQ}_1$, we compute and report the overlap (in percentage) of authors that used the various communication channels.

Similarly, for $\mathbf{RQ}_2$, we compute the overlap (in percentage) of links existing between different authors when considering different sources of information.

Besides such a quantitative analysis of the links, we are also interested to investigate the *nature* of the discussions occurring over the different communication channels. Undoubtedly, the most suitable way to do this kind of analysis is to rely on grounded theory, as done by Guzzi *et al.* [12]. However, This is not feasible when analyzing several sources from seven projects. Instead, we perform two different kinds of analyses. First, we perform a quantitative analysis, done using topic models. For each project and for each communication channel, we build a topic model using Latent Dirichlet Allocation (LDA) [13]. LDA allows to fit a generative probabilistic model from the term occurrences in a corpus of documents. Basically each document is treated as a probability distribution of topics, in turn being distributions of words. The corpus for emails and issues consists of message subjects/bug title/short descriptions only (each of them represents a document in the corpus), because the rest of the message/issue discussion often contains details that would only add noise to the overall topic characterization. For IRC discussions, we took all messages (each of them is a document), since they are often very short and because no subject/short title is available in this case. The corpus is then processed by applying English stop word removal and Snowball stemming, and then topic models are generated. After, we analyze how topics discussed over the various communication channels are similar, by comparing the topic models using the Hellinger distance [14]. Since the Hellinger distance varies between 0 and 1, and since

we were interested to show the similarity of the discussion between pairs of communication channels, we convert it into a similarity as follows $S(P,Q) = 1 - H(P,Q)$. The whole topic analysis has been performed using the *topicmodels* package of the *R* statistical environment. Note that, when applying LDA one needs to calibrate the number of topics k, the smoothing factors for topic distributions in documents (α) and word distributions in topics (β), and the number of Gibbs iterations (n) required to generate the topic model. Although we are aware that LDA can produce sub-optimal results if not properly calibrated [15], in this case we did it by observing how our results vary by considering a number of topic $k \in \{25, 50, 100, 200\}$. For all projects, we did not notice any substantial difference when going beyond 50. For this reason, we have set $k = 50$. Similarly, we set $\alpha = 0.1$, and $\beta = 1/k$, and $n = 10$.

To address $\mathbf{RQ}_3$, we use the communication links extracted from the different sources of information to (i) recommend developers playing particular roles, and (ii) replicate the results of the study reported by Bird *et al.* [5]. Specifically, with respect to point (i) we rank developers using the following metrics:

- **Degree**: *i.e.*, the number of in-out communication links a developer has within a given communication channel [16]. The conjecture is that a person taking the leadership in a discussion would have a high degree. Degree metrics have been computed using the *R* package *sna*. To understand whether high-degree developers identified by the various communication networks are actually recognized as "important" developers by the community, we rely on the Ohloh[1] *Kudos* score. A *Kudos* depends on the level of appreciation or respect of a developer working for a project receives, and it is based on the judgement of other project members[2]. Specifically, a member can give *Kudos* to other members, by assigning them a score ranging between 1 and 10. An example of *Kudos* ranking for the project Apache HTTPD can be found at the URL http://www.ohloh.net/p/apache/contributors.
- **Mentorship**: a project member is a mentor if s/he shows the ability to effectively train other people, generally newcomers. While the identification of developers with high degree is quite trivial, to identify mentors we rely on a recommender system defined by Canfora *et al.* [8]. This approach is able to identify, given a newcomer joining the project in a given moment, the project's member that has been her/his mentor by taking into account factors like (i) the communication exchange between the newcomer and each project member, (ii) the level of sociability (degree) of each project member, and (iii) the difference in seniority of the newcomer with each project member. The previously performed empirical evaluation indicated a 75% accuracy in the mentorship identification [8].

Note that the aforementioned *degree* and *mentorship* metrics are not Boolean, they rather indicate to what extent a project member plays (or not) one of the two roles described above.

As for point (ii), Bird *et al.* [5] exploit the social network built by mining mailing lists to compute the Spearman's rank correlation [17] between the number of changes (commits) performed by developers on source code files (*srcChanges*), on documentation changes (*docChanges*) and their out-degree (*i.e.*, to how many different project members a developer sends messages), in-degree (*i.e.*, from how many different project members a developer receives messages), and betweenness (*i.e.*, an indication of the extent to which a developer is in communication paths involving other developers [16]). We replicate such a study with the four sources of information considered here, however without making distinction between in- and out-degree—because this is in principle not possible in channels such as chat—but using the overall degree metric instead.

TABLE II. RQ_1: OVERLAP (IN PERCENTAGE) BETWEEN AUTHORS CONTRIBUTING TO DIFFERENT SOURCES. *cc* ≡ ISSUES ∪ EMAILS ∪ CHAT.

Apache HTTPD					
	#authors	issues	chat	emails	cc
commits	45	80%	6%	80%	91%
issues	36		8%	86%	100%
chat	3	100%		100%	100%
emails	36	86%	8%		100%
cc	41	88%	7%	88%	
CXF					
	#authors	issues	chat	emails	cc
commits	21	57%	71%	43%	76%
issues	12		92%	58%	100%
chat	15	73%		53%	100%
emails	9	78%	88%		100%
cc	16	75%	94%	56%	
Hibernate					
	#authors	issues	chat	emails	cc
commits	77	24%	42%	34%	56%
issues	19		68%	68%	100%
chat	32	41%		59%	100%
emails	26	50%	73%		100%
cc	43	44%	74%	60%	
Infinispan					
	#authors	issues	chat	emails	cc
commits	40	73%	78%	75%	90%
issues	29		90%	83%	100%
chat	31	84%		87%	100%
emails	30	80%	90%		100%
cc	36	81%	86%	83%	
Lucene					
	#authors	issues	chat	emails	cc
commits	32	78%	53%	31%	84%
issues	25		64%	36%	100%
chat	17	94%		41%	100%
emails	10	90%	70%		100%
cc	27	92%	63%	37%	
Samba					
	#authors	issues	chat	emails	cc
commits	101	79%	21%	71%	90%
issues	80		25%	76%	100%
chat	21	95%		86%	100%
emails	72	84%	25%		100%
cc	91	88%	23%	79%	
Weld					
	#authors	issues	chat	emails	cc
commits	66	45%	32%	3%	52%
issues	30		56%	0%	100%
chat	21	81%		9%	100%
emails	2	0%	100%		100%
cc	34	88%	62%	5%	

D. Replication Package

The replication package for this study is publicly available[3]. Specifically, we provide: (i) the downloaded data from all sources of all projects, (ii) the developers' links extracted, and (iii) the *R* scripts and working data sets used to produce the results reported in this paper.

III. ANALYSIS OF THE RESULTS

This section discusses the results achieved in our study and aimed at addressing the three research questions formulated in Section II-A.

[1] http://www.ohloh.net

[2] http://meta.ohloh.net/kudos

[3] www.rcost.unisannio.it/mdipenta/devel-net.tgz

A. RQ_1: To what extent do developers discuss through the different communication channels?

Table II reports (i) the number of developers (*i.e.*, commit authors) contributing to the different sources of information; and (ii) the percentage of overlap between the different sources. In addition, we also considered the union of all communication channels (emails, issues, and chat). It is important to note that given a source S_i indicated on the row and a source S_j indicated on the column, the overlap of the authors $A(S_i)$ participating in S_i with the authors $A(S_j)$ participating in S_j is given by $|A(S_i) \cap A(S_j)|/|A(S_i)|$. For this reason Table II is not symmetric.

First, we can notice that a good percentage of commit authors were found in the communication channels (column *cc*): such a percentage varies between 52% for WELD and 90% for SAMBA, with an average value of 75%. The communication channel attracting the largest percentage of authors varies between projects. For three projects (HIBERNATE, INFINISPAN, and CXF) the most popular channel is the chat, for SAMBA, LUCENE and WELD it is the issue tracker, while for HTTPD are issues and emails.

In six projects out of seven (*i.e.*, all but INFINISPAN), authors mainly use two out of three communication channels, whereas the third one is only used sporadically. For example, in Samba the issue tracker and the emails are used by 79% and 71% of the authors respectively, while only 21% use the chat. There may be many factors, such as the project size, internal organization and structure, or its age, that may influence the proneness of developers to use different communication sources. For example, SAMBA is relatively older than other projects (16 years of life, since 1998) and developers used for years mailing lists to communicate. Only recently, they also adopted an issue tracker and, very recently, developers began to systematically use the chat. Indeed, the set of authors that exchange messages over issue tracker and mailing lists largely overlaps with the (small) set of people using the chat (95% and 86% respectively). However, not all old projects have such a behavior. Consider, for example, LUCENE. This is a relatively old project (2000), however developers mainly rely on chat and issue trackers to exchange messages and organize their work. This also confirm what Guzzi *et al.* [12] found when analyzing its mailing lists. LUCENE, indeed, differs from SAMBA in terms of number of authors (32 vs. 101) and domain (it is more a scientific project than a widely-used utility like SAMBA). In some sense, developers form a sort of "small community" that tends to gather a lot over the chat.

In HTTPD the IRC is poorly used by developers, while it is the most used communication channel in HIBERNATE, where developers began to use IRC just two years after the project started. INFINISPAN is also a relatively young project (2008), and in this case the use of all communication channels is very balanced: 73% for the issue tracker, 78% for the chat and 75% for emails. Last, but not least, WELD authors very rarely use emails during the observed period. That is, developers find it more convenient to directly interact through chat or to discuss specific issues over proper means, *i.e.*, the issue tracker.

RQ_1 Summary: It is unlikely that all developers communicate over all channels, therefore to properly observe their interaction multiple channels should be considered. In addition, for the projects under study, while in the past developers used emails as main communication channel, nowadays they are massively using chats or issue trackers.

TABLE III. RQ_2: NUMBER OF AUTHOR LINKS FOUND IN THE DIFFERENT SOURCES OF INFORMATION, AND OVERLAP (IN PERCENTAGE) BETWEEN THEM.

Apache HTTPD						
	#links	commits	issues	chat	emails	cc
commits	371		13%	0%	19%	29%
issues	100	49%		0%	26%	100%
chat	0	0%	0%		0%	0%
emails	195	37%	13%	0%		100%
cc	269	41%	37%	0%	72%	
CXF						
	#links	commits	issues	chat	emails	cc
commits	73		14%	26%	5%	38%
issues	30	33%		40%	13%	100%
chat	85	22%	14%		2%	100%
emails	11	36%	36%	18%		100%
cc	109	26%	28%	78%	10%	
Hibernate						
	#links	commits	issues	chat	emails	cc
commits	184		3%	1%	12%	21%
issues	19	26%		16%	26%	100%
chat	248	8%	1%		13%	100%
emails	81	28%	6%	41%		100%
cc	307	13%	6%	81%	26%	
Infinispan						
	#links	commits	issues	chat	emails	cc
commits	193		33%	36%	22%	63%
issues	147	43%		37%	29%	100%
chat	445	16%	12%		19%	100%
emails	165	26%	25%	50%		100%
cc	593	20%	25%	75%	100%	
Lucene						
	#links	commits	issues	chat	emails	cc
commits	195		19%	11%	4%	27%
issues	140	27%		29%	4%	100%
chat	110	20%	37%		5%	100%
emails	23	30%	26%	22%		100%
cc	222	23%	63%	50%	10%	
Samba						
	#links	commits	issues	chat	emails	cc
commits	729		16%	2%	13%	27%
issues	360	33%		1%	18%	100%
chat	50	28%	10%		18%	100%
emails	313	30%	21%	3%		100%
cc	647	31%	56%	8%	48%	
Weld						
	#links	commits	issues	chat	emails	cc
commits	82		5%	16%	0%	18%
issues	24	17%		38%	0%	100%
chat	109	12%	8%		0%	100%
emails	0	0%	0%	0%		0%
cc	124	12%	19%	88%	0%	

B. RQ_2: How do the inferred links between developers overlap when using different sources of information?

Table III reports the number of authors' links found in the different sources of information, and the overlap (in percentage) between the various sources (plus the union of all communication channels *cc*). A link represents a pair of authors that interact within a source. In most cases, the sources exhibiting the highest number of links are issue trackers and chat logs. A strong exception to this trend is the IRC chat of HTTPD, only used by three developers that used it just to communicate with other people external to the development team (as pointed out, in the user support page of HTTPD[4])

The links identified from the commits have an overlap with other sources ranging between 0% (commits vs. emails in WELD) and 36% (commits and chat in INFINISPAN). Clearly, the former 0% is due to the limited participation of authors in WELD mailing lists as observed in **RQ_1**. When computing the overlap in the opposite direction (other sources vs. commits), we can notice relatively high values for issues (49% HTTPD, 43% INFINISPAN, 33% CXF and SAMBA, 27% LUCENE, 26% HIBERNATE). If merging all communication channels, the link overlap of commits with other sources raises, going from 18% for WELD up to 63% for INFINISPAN. This highlights

[4]https://httpd.apache.org/support.html

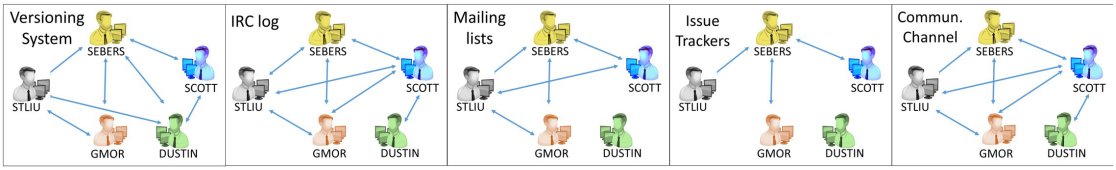

Fig. 1. Hibernate: network of five developers as it is captured from different sources of information.

the importance of analyzing more than one communication channel when building developers collaboration networks.

However, the amount of messages, and thus, links in channels like the chat and issue tracking system are in general higher than messages and links in emails and IRC logs. This is partially due to the strong assumption for chat and issue tracking system that all people participating to a discussion are considered linked (especially for the chat where the number of links is very high). Indeed, a developer in such channels, very often answers the previous comment in a discussion, and thus, he/she replies (communicates) with few people in a discussion. For the emails this cannot happen because, it is a point-to-point communication. Thus, merging links between different sources of information requires a particular attention (a single link between developers in emails is more reliable with respect to a link in chat). It is interesting to note from Table III that for the majority of the projects, links from emails have a higher overlap with links in issue than links in chat. This means that communication in mailing lists (more reliable) are intrinsically more bound to the (one-to-many) communication of the issue tracker.

In CXF, we can notice that the overlap between chat and emails is very low (2%, whereas the opposite is 18%), while it raises up to 40% between issues and chat. In HIBERNATE and INFINISPAN the highest overlap is between emails and chat (41% and 50% respectively). In LUCENE, both issue tracker and chat have a limited overlap with mailing lists (4% and 5%).Instead, the overlap between chat and the issue tracker is 37% (reverse 29%). The overlap between links in the issue tracker and chat is also relatively high in WELD (38%) where the reverse overlap is however low (8%), that is, there are many links in the chat that do not appear in the issue tracker. Finally, as also mentioned in **RQ**$_1$, SAMBA developers are less prone to use the chat, and this explains its limited overlap with mailing lists and the issue tracker.

Let us consider, for example, the subset of five HIBERNATE developers depicted in Figure 1. The figure shows five different networks built considering the four sources of information considered in the study and their combination. When considering only a source of information, some links may be missing: for example *Dustin* performs commit with others, but he talks with them only on the chat.

As also noticed by Shihab *et al.*[18], IRC online meetings are often planned to answer questions related to common project topics, or for brainstorming. For example, during an IRC meeting a very active author of HIBERNATE wrote: *"is there a better way? dunno like I said this is brainstorming and I have not given lots of thought to these cases"*. Another author said: *"but we also need to create the attributes and values in the entity binding.."*. Topics that are also often discussed on the IRC are related to planning testing activities *"however a pure standalone test suite would make things easier..."*. Also, developers discuss how to prioritize activities on issues and whether or not to open issues on the issue tracker *"okay I think it is a bug and I'm going to create a jira first"*.

TABLE IV. SIMILARITY MEASURE OF TOPICS EXTRACTED FROM DIFFERENT COMMUNICATION CHANNELS.

	issues vs. emails	issues vs. chat	emails vs. chat
Apache HTTPD	0.17	0.09	0.06
CXF	0.86	0.11	0.01
Hibernate	0.11	0.02	0.03
Infinispan	0.07	0.03	0.03
Lucene	0.08	0.3	0.02
Samba	0.06	0.02	0.02
Weld	0.11	0.04	0.03

By applying topics model as described in Section II-C, the words describing the topic with the highest probability of chats (for HIBERNATE) are *test, fix, plan, project, unresolved, migration, integration, branch*. Instead, the topic with the highest probability for the issue tracker contains *fail, error, test, issue, broken, valid, wrong, delete, build, core*, while the emails have *test, build, valid, core, api, branch, fail, error, build, documentation, strategies*.

Table IV reports the similarity (computed using the Hellinger distance) of all channel pairs. One can notice that values in the first column (issues vs. emails) are always higher than those in the other two columns, where issues and emails are compared with the chat. Among other cases, one can notice the very high similarity in CXF between issues and emails (0.86). For this project, we noticed that the top topics for issues and emails share several words such as *test, build, valid, core, fail, error, doc, strategies*. Recently, developers are using issue trackers more and more as a valid alternative to mailing lists to discuss various kind of issues, not only related to specific bugs to fix or features to add/improve. Vice versa, the IRC chat has intrinsically a more interactive nature, and thus it is more suitable to brainstorming.

RQ$_2$ **summary:** The overlap of communication links between various sources is relatively low (generally below 30%-40%) and varies depending on the project. Therefore, data from multiple channels should be merged to have a better view of developers' interactions. The topics (and links) being discussed in issues and emails are closer to each other than those discussed in the IRC chat.

C. *RQ*$_3$: *How do social network metrics change when using different sources, and how would this impact on using such information to build recommenders?*

In the following we report results of **RQ**$_3$ for what concerns (i) identifying high-degree developers and mentors, and (ii) studying the correlation between social roles and change activities.

1) Recommending Coordinators and Mentors: Table V reports the percentages of overlap between the top five *high-degree* authors, and *mentors* for the different sources of information. Note that we did not identify mentors from the

TABLE V. RQ_3: PERCENTAGE OF OVERLAP BETWEEN TOP FIVE *Coordinators* AND *Mentors* AS EXTRACTED FROM THE FOUR SOURCES OF INFORMATION.

HTTPD	**[Coordinators]**					**[Mentors]**		
	issues	chat	emails	cc	kudos	chat	emails	cc
commits	40%	0%	20%	0%	20%	-	-	-
issues		0%	60%	0%	20%	20%	60%	20%
chat			0%	0%	0%		20%	80%
emails				0%	**60%**			20%
CXF	**[Coordinators]**					**[Mentors]**		
	issues	chat	emails	cc	kudos	chat	emails	cc
commits	40%	20%	40%	20%	40%	-	-	-
issues		20%	40%	20%	20%	40%	60%	60%
chat			20%	100%	20%		20%	40%
emails				20%	**60%**			60%
Hibernate	**[Coordinators]**					**[Mentors]**		
	issues	chat	emails	cc	kudos	chat	emails	cc
commits	40%	40%	40%	60%	20%	-	-	-
issues		40%	40%	20%	60%	20%	40%	40%
chat			40%	80%	40%		20%	20%
emails				60%	**60%**			60%
Infinispan	**[Coordinators]**					**[Mentors]**		
	issues	chat	emails	cc	kudos	chat	emails	cc
commits	80%	40%	80%	80%	40%	-	-	-
issues		40%	80%	80%	40%	20%	60%	60%
chat			40%	60%	0%		20%	20%
emails				80%	**60%**			100%
Lucene	**[Coordinators]**					**[Mentors]**		
	issues	chat	emails	cc	kudos	chat	emails	cc
commits	40%	0%	20%	80%	**60%**	-	-	-
issues		20%	0%	60%	40%	40%	20%	60%
chat			0%	20%	20%		20%	40%
emails				20%	**60%**			40%
Samba	**[Coordinators]**					**[Mentors]**		
	issues	chat	emails	cc	kudos	chat	emails	cc
commits	60%	20%	80%	80%	**80%**	-	-	-
issues		0%	60%	60%	60%	20%	60%	80%
chat			20%	40%	20%		40%	40%
emails				80%	**80%**			80%
Weld	**[Coordinators]**					**[Mentors]**		
	issues	chat	emails	cc	kudos	chat	emails	cc
commits	60%	0%	0%	20%	0%	-	-	-
issues		0%	0%	20%	**20%**	40%	20%	60%
chat			0%	80%	0%		20%	60%
emails				0%	**20%**			40%

commits as this does not make sense. In addition, in Table V we also report, for each source of information, the percentage of overlap between the top five *high-degree* authors and the top five developers that obtained the highest *Kudos scores* (column *kudos* in Table V).

In terms of *degree*, the percentage of overlap between the different sources is low (36%, on average). In 68% of the cases the overlap between the compared pairs of sources is $\leq 40\%$, and just in 20% of the cases the overlap is $\geq 80\%$. Vice versa, when recommending *mentors*, the average overlap between all pairs of sources is 41%, in 67% of cases the overlap is $\leq 40\%$, while just in 9% of cases it is $\geq 80\%$. However, as highlighted by the results of the RQ_2, topics (and links) discussed in mailing lists are closer to the topics (and links) discussed in issue tracker. Thus, ideally, if we focus the attention between these two communication channels we expect to have a higher overlap in terms of coordinators and mentors. This result is confirmed in Table V. In particular, the percentage of overlap of the coordinators between emails and issues is 40% on average, and the percentage in terms of mentors is 47% on average. Vice versa, the chat obtained the lower overlap of mentors/coordinator with the other communication channels (it is very often lower than 20%). If we do not consider the chat, in terms of *degree*, the percentage of overlap between the different sources increases from 36% to 46% (on average), while in terms of *mentors*, the percentage of overlap between the different sources increases from 41% to 47% (in average). Thus, it is clear that the chat channel identifies a set of mentors/coordinators that are very decoupled with the set of mentors/coordinators provided by the other communication channel. The overlap of the top *Kudos* developers with those having the highest degree highlights such a finding. By looking at Table V, it is evident that the lowest overlap between top *degree* and top *Kudos* is obtained by the chat channel, while the highest overlap is achieved if considering emails. This means that, by computing high *degree* on the network obtained from emails, we are able to identify developers that have a high reputation in the project.

TABLE VI. RQ_3: HIBERNATE'S TOP FIVE PROJECT MEMBERS: HIGH-DEGREE DEVELOPERS AND MENTORS.

Coordinators						
Rank	commits	issues	chat	emails	cc	kudos
1	sebers	emmanuel	sanne	sebers	sebers	gavin
2	stliu	hardy	scott	sanne	sanne	sebers
3	lukasz	gmorling	gail	hardy	gail	emmanuel
4	bmeye	bmeye	emmanuel	emmanuel	scott	hardy
5	gail	sebers	sebers	stliu	stliu	erik
Mentors						
Rank	commits	issues	chat	emails	cc	
1	-	bmeye	bein	sebers	sebers	
2	-	hardy	pmui	emmanuel	scott	
3	-	lukasz	suppor	max	sanne	
4	-	emmanuel	adnan	hardy	hardy	
5	-	sanne	stuartdou	sanne	stliu	

In summary, recommendations of high-degree developers and mentors computed using developers collaboration networks mined from different sources can be different. The set of mentors/high-degree developers identified relying on chat is very decoupled with the set of mentors and high degree developers identified by the other channels. This analysis is also confirmed by the analysis of *Kudos*.

Table VI reports the top five *high-degree* developers and *mentors* extracted from each source of information of HIBERNATE. If one is interested in knowing which are the *high-degree* developers of the HIBERNATE project, she could choose to mine any of the available sources of information, achieving however different results case by case. Indeed, the project's author *sebers* is the only one identified as *high-degree* developer in all cases, while substantial differences can be observed for other authors. An interesting case is related to the HIBERNATE developer *sanne*, that has been identified as a coordinator when mining chat, emails, or the union of all communication channels (*cc*), while he is not in the top five when mining commits (he is a committer, nevertheless) and issues. His Linkedin profile[5] mentions that he is one of the project's members leading HIBERNATE. However, if for instance one limits the collaboration/communication analysis to commits and issues, this information would not emerge.

2) Studying the correlation between developers activity and social network metrics [5]: Tables VII and VIII show the results we achieved on HTTPD and HIBERNATE[6] when replicating the study by Bird *et al.* [5] aimed at analyzing the correlation between changes performed by developers on source code (*srcChanges*) and on documentation (*docChanges*) with two social network metrics highlighting the importance of a developer in the social network (*i.e.*, degree, and betweenness—see Section II-C). Bird *et al.* [5] perform their study on Apache HTTPD by relying on its mailing list to build the developers social network. Their results show a high correlation between the analyzed social network metrics and changes on source code performed by developers, indicating that developers who actually commit changes, play much more significant roles in the email community than non-developers [5]. For HTTPD, the replication of their study led us to similar results both when using mailing list and the issue tracker as sources to build the social network (see Table VII). Note that for HTTPD it was not possible to exploit information derived from the IRC chat, given the absence of links between

[5]http://it.linkedin.com/in/sannegrinovero

[6]Results for the other systems are available in our replication package.

TABLE VII. APACHE HTTPD: CORRELATION BETWEEN THE TOTAL NUMBER OF CHANGES, CHANGES TO SOURCE, CHANGES TO DOCUMENTS, DEGREE, AND BETWEENNESS.

EMAILS					
	changes	srcChanges	docChanges	degree	betweenness
changes	1	0.58	0.64	0.37	0.40
srcChanges		1	0.35	0.44	0.41
docChanges			1	0.48	0.52
degree				1	0.81
betweenness					1
Issues					
	changes	srcChanges	docChanges	degree	betweenness
changes	1	0.50	0.66	0.28	0.30
srcChanges		1	0.44	0.55	0.56
docChanges			1	0.40	0.48
degree				1	0.78
betweenness					1

TABLE VIII. HIBERNATE: CORRELATION BETWEEN THE TOTAL NUMBER OF CHANGES, CHANGES TO SOURCE, CHANGES TO DOCUMENTS, DEGREE, AND BETWEENNESS.

EMAILS					
	changes	srcChanges	docChanges	degree	betweenness
changes	1	0.98	0.54	0.62	0.52
srcChanges		1	0.52	0.60	0.50
docChanges			1	0.45	0.39
degree				1	0.88
betweenness					1
ISSUES					
	changes	srcChanges	docChanges	degree	betweenness
changes	1	1	0.61	0.29	0.68
srcChanges		1	0.61	0.19	0.68
docChanges			1	0.33	0.53
degree				1	0.68
betweenness					1
CHAT					
	changes	srcChanges	docChanges	degree	betweenness
changes	1	0.98	0.56	0.24	0.24
srcChanges		1	0.54	0.25	0.27
docChanges			1	0.06	0.10
degree				1	0.78
betweenness					1

developers in such a communication channel (see Table III). Thus, on this system, the conclusions drawn on the correlation between code changes and social network metrics do not change across different communication channels.

The situation is different for HIBERNATE. In this case, we can observe a high correlation between *srcChanges* and the considered social network metrics when considering the mailing lists as communication channel. When considering the issues, a high correlation can only be observed between betweenness and *srcChanges*. There is no correlation when building the social network from IRC communication (see Table VIII). We achieved results similar to HTTPD also for INFINISPAN and LUCENE, while results inline with HIBERNATE have been observed for CXF, SAMBA, and WELD. In summary, social network metrics captured from mailing lists and issue tracker reflect well the developers' activity, while this is not the case for the chat.

RQ_3 summary: Social network studies and recommenders in software engineering should not limit their information mining to a single source. However, some social network metrics extracted from the different sources may have a different interpretation, e.g. high degree on chat does not necessarily correspond to high code change activity.

IV. THREATS TO VALIDITY

Construct validity threats concern the relationship between theory and observation. Such threats are mainly due to imprecision in the mapping of names used in different sources, and in how links were identified. As for the unification/mapping of names, as explained in Section II-B we have used an approach inspired from previous work [5], [19], [8] and complemented it by a thorough manual validation. However, we cannot exclude possible mistakes. Nevertheless, given the high number of developers involved in the study, it is unlikely that small deviations will change the essence of our findings.

Concerning the identification of links, we used state-of-the-art approaches to identify links in mailing lists, issue trackers and chats. However, we are aware that the participation to an issue in issue trackers does not mean communicating with everybody involved there, and similarly it is likely that not everybody in a chat session is really involved in each specific discussion. Finally, we are aware that links inferred from versioning system may have little value because people working on the same file might never get in touch. Nevertheless, our aim is to show that links extracted from code changes or from communication channels, although overlapped, have different meaning and therefore can be quite different.

Last, but not least, it is important to point out that in this study we did not aim at validating the mining links (which might be part of our future work), because we were interested to only understand how the mined communication links vary between sources and how do such links influence studies conducted upon such datasets.

Threats to *internal validity* concern factors that could have influenced our results. Our study is based on what in our opinion are the most widely used communication channels in open source projects. As it will be discussed in Section V, other channels—*e.g.*, microblogging through Twitter—indeed exist. While we found that for the analyzed projects Twitter is mainly used for advertisement purposes, in a different setting—*e.g.*, small industrial organization—it could be used during development. Last, but not least, besides all (written) sources of information one can consider, we are aware that there is still a portion of the developers' communication happening by voice, and that are not traceable elsewhere [2].

External validity threats concern the generalizability of our results. The study is limited to seven systems and, for consistency and comparison between projects, to the most recent project years. Although we expect similar findings, further, larger studies need to be conducted to generalize, confirm, or contradict our findings.

V. RELATED WORK

In the following, we discuss work concerning the analysis of developers collaboration networks (DCN) for various purposes in the context of software engineering studies, and with the aim of building software engineering recommenders.

Previous studies analyzed DCN applying social network analysis on data extracted from Versioning Systems [20], [21], [10], [11], [22], [23], [24] community at SourceForge, finding that the obtained developer network is a scale-free network. For example, Pohl *et al.* [11] showed how social networks could be used to determine roles in the community of developers belonging to the a software project. We share with Pohl *et al.* the approach used to identify relations between developers from versioning system data (two developers are connected if contributed to the same file during the same period). Studies by Singh *et al.* [22] observed how committers networks is a small-world network. Surian *et al.* [23] findings are consistent with

those of Singh *et al.* [22]; that is, the small-world phenomenon also exists in SourceForge, especially when developers in a network are separated, on average, by approximately 6 hops. More recently, Meneely *et al.* [21] used two issue tracking annotations—*i.e.*, solution originator and solution approver—from bug databases to complement the developers network of versioning data. In a subsequent work, Meneely *et al.* [10] showed that SNA metrics represent socio-technical relationships in open source development projects. This reflects the work done in our $\mathbf{RQ}_3$, which however highlights that such socio-technical relationships may change when using different sources of information.

Various authors have investigated developers' collaboration through mailing lists [5], [25], [12], [26]. Bird *et al.* [5] discovered that—in mailing list DCN—few members account for a large proportion of messages sent and of replies. They also found high correlations between various social network status metrics and source code development. Bird *et al.* [3] analyzed the relationship between communications structure and code modularity, and found that sub communities identified using communication information are related to code collaboration behavior. Sometimes, mailing list communication cross the boundaries of a single project as studied by Canfora *et al.* [19] on the collaboration between OpenBSD and FreeBSD developers with the aim of fixing related bugs. The heterogeneity of email content and discussion was investigated by Bacchelli *et al.* [25] and Guzzi *et al.* [12]. Bacchelli *et al.* [25] presented a technique that classifies email lines into five categories (text, junk, code, patch, and stack trace) and evaluated such approach on a (statistically) significant amount of emails gathered from mailing lists of four unrelated open source systems. Guzzi *et al.* [12] quantitatively and qualitatively analyzed a sample of 506 email threads from the development mailing list of Apache Lucene. Their study shows that developers participate in less than 75% of the threads, and that in only about 35% of the threads source code details are discussed.

Hence, developers also discuss through other communication channels, including issue trackers and IRC. Indeed, IRC meetings are increasing in popularity among OSS developers [27]. Elliot *et al.* [28] reveal how, using IRC instant messaging streams, persistent IRC logs and mailing lists help not only to build a community but also resolve conflicts. Shihab *et al.* [18] analyzed IRC logs and found that a small and stable number of the participants contribute the majority of messages. LaToza *et al.* [29] surveyed eleven developers with the aim of investigating common practices and their satisfaction in software development. They discovered several barriers preventing email (and in general written communication) usage. They found that face-to-face communication has advantages and that the use of more interactive communication channels (like IRC) is more desirable than emails.

While mailing lists have been used a lot in the past, nowadays many projects are moving most of the discussion onto issue trackers, that are used besides the simple discussion of bugs to be fixed. For this reason, various authors have proposed developers based on issue trackers. Haythornthwaite [30] found that the set of core developers identified considering interactions on issue trackers differ from the "formal" lists of contributors published on projects' Website. Hong *et al.* [4] compared the evolution of DCN extracted from issue trackers with the evolution of general social networks (*e.g.*, Facebook or Twitter, etc.), finding some commonalities and differences. Other works by Crowston *et al.* [31] and Zhou *et al.* [32] used co-occurrence of developers on bug reports as indicators of a social link. With the aim at addressing the problem of inter-team coordination, Begel *et al.* [33] presented Codebook, a framework for connecting engineers and their work artifacts together.

Recently, several researchers investigated and evaluated the the role played by communications in Twitter and more in general, the role played by "microblogging", in software development organizations [34], [35], [36], [37]. Zhao *et al.* [37] surveyed 11 microblog participants to better understand the conversational aspects of Twitter discovering the potential benefits it brings to informal communication at work. However, as Zhang *et al.* [36] highlighted, there is a large variation in the posting activity of various users, and there are *barriers* in adopting such new social communication channels. Moreover, Ehrlich *et al.* [34] showed how different the use of the external/internal microblogs are: external microblogs are used for sharing general information; instead, internal microblogs are used to technical assistance and discussion. Finally, Dullemond *et al.* [35] evaluated microblogging discussions, and found how "mood-activity environment" helps to obtain information that is traditionally harder to obtain in a less volatile form. In summary, although there are barriers, microblogging could likely become another promising communication channel. However, we did not consider it in our study, because (i) we found that the Twitter accounts of the analyzed projects are mainly used for advertisements, *e.g.*, of new releases; (ii) since we deal with (sometimes large) open source projects rather than closed organizations, it is not feasible to keep track of the Twitter accounts of all developers (if any).

VI. Conclusion and Future Work

In this paper we analyzed developers' communication over different channels (mailing lists, issue trackers, IRC chat) and their co-change activities captured from versioning systems. The study concerned a period of observation of at least two years for seven open source projects.

Results of the study highlighted that analyzing developers collaboration/communication through specific channels would only provide a partial view of the reality, and that different channels may provide different perspectives of developers' communication. In particular, (i) not all developers use all communication channels; and (ii) people mainly interact through two out of three communication channels, whereas the third one is only used sporadically.

Therefore, if using specific collaboration/communication networks for various purposes—*e.g.*, identifying experts or mentors—one should be careful as different channels may lead to more or less accurate—and in any case different—results. For example, we found that high degree in chat does not necessarily correspond to high code change activity, while for mail and issue it is correlated. Thus, especially when such networks are used to identify development high degree, the choice of the most appropriate source should be done carefully, bearing in mind what was the purpose of such a channel in the project (*e.g.*, whether or not it was used to coordinate coding activities).

Work-in-progress aims at replicating the study on further projects, and also at showing how the result of other applications of developers' social network analysis change when using different sources. Last, but not least, we plan to survey developers of the analyzed projects to collect and analyze their perception about the strength of the identified communication links.

REFERENCES

[1] F. Brooks, *The Mythical Man-Month 20th anniversary edition*. Boston, MA, USA: Addison-Wesley, 1995.

[2] J. Aranda and G. Venolia, "The secret life of bugs: Going past the errors and omissions in software repositories," in *31st International Conference on Software Engineering, ICSE 2009, May 16-24, 2009, Vancouver, Canada, Proceedings*, 2009, pp. 298–308.

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

[4] Q. Hong, S. Kim, S. C. Cheung, and C. Bird, "Understanding a developer social network and its evolution," in *IEEE 27th International Conference on Software Maintenance, ICSM 2011, Williamsburg, VA, USA, September 25-30, 2011*. IEEE, 2011, pp. 323–332.

[5] C. Bird, A. Gourley, P. Devanbu, M. Gertz, and A. Swaminathan, "Mining email social networks," in *Proceedings of the 2006 international workshop on Mining software repositories*, ser. MSR '06. New York, NY, USA: ACM, 2006, pp. 137–143.

[6] N. Bettenburg and A. E. Hassan, "Studying the impact of social structures on software quality," in *International Conference on Program Comprehension, ICPC 2010*, 2010, pp. 124–133.

[7] A. Kumar and A. Gupta, "Evolution of developer social network and its impact on bug fixing process," in *Proceedings of the 6th India Software Engineering Conference*. ACM, 2013, pp. 63–72.

[8] G. Canfora, M. Di Penta, R. Oliveto, and S. Panichella, "Who is going to mentor newcomers in open source projects?" in *Proceedings of the 20th ACM SIGSOFT Symposium on the Foundations of Software Engineering*, Cary, NC, USA, 2012, p. 44.

[9] S. Panichella, G. Canfora, M. Di Penta, and R. Oliveto, "How the evolution of emerging collaborations relates to code changes: an empirical study," in *International Conference on Program Comprehension, ICPC 2014*, 2014.

[10] A. Meneely and L. Williams, "Socio-technical developer networks: Should we trust our measurements?" in *Proceedings of the 33rd International Conference on Software Engineering*. New York, NY, USA: ACM, 2011, pp. 281–290.

[11] M. Pohl and S. Diehl, "What dynamic network metrics can tell us about developer roles," in *Proceedings of the 2008 international workshop on Cooperative and human aspects of software engineering*, ser. CHASE '08. New York, NY, USA: ACM, 2008, pp. 81–84.

[12] A. Guzzi, A. Bacchelli, M. Lanza, M. Pinzger, and A. van Deursen, "Communication in open source software development mailing lists," in *Proceedings of the 10th Working Conference on Mining Software Repositories, MSR '13, San Francisco, CA, USA, May 18-19, 2013*. IEEE / ACM, 2013, pp. 277–286.

[13] D. M. Blei, A. Y. Ng, and M. I. Jordan, "Latent Dirichlet Allocation," *J. Mach. Learn. Res.*, vol. 3, pp. 993–1022, March 2003.

[14] M. Nikulin, "Hellinger distance," *Encyclopedia of Mathematics*, 2001.

[15] A. Panichella, B. Dit, R. Oliveto, M. Di Penta, D. Poshyvanyk, and A. De Lucia, "How to effectively use topic models for software engineering tasks? an approach based on genetic algorithms," in *35th IEEE/ACM International Conference on Software Engineering, ICSE 2013*, San Francisco, CA, USA, May 18-26, 2013, pp. 522–531.

[16] J. P. Scott, *Social Network Analysis: A Handbook (2nd edition)*. Sage Publications Ltd, 2000.

[17] J. H. Zar, "Significance testing of the spearman rank correlation coefficient," *Journal of the American Statistical Association*, vol. 67, no. 339, pp. pp. 578–580, 1972.

[18] E. Shihab, Z. M. Jiang, and A. Hassan, "Studying the use of developer irc meetings in open source projects," in *Software Maintenance, 2009. ICSM 2009. IEEE International Conference on*, 2009, pp. 147–156.

[19] G. Canfora, L. Cerulo, M. Cimitile, and M. Di Penta, "Social interactions around cross-system bug fixings: the case of FreeBSD and OpenBSD," in *Proceedings of the 8th International Working Conference on Mining Software Repositories, MSR 2011, Waikiki, Honolulu, HI, USA, May 21-28, 2011*, 2011, pp. 143–152.

[20] A. Capiluppi and M. Michlmayr, "From the cathedral to the bazaar: An empirical study of the lifecycle of volunteer community projects," in *Open Source Development, Adoption and Innovation*, International Federation for Information Processing. Springer, 2007, pp. 31–44.

[21] A. Meneely, M. Corcoran, and L. Williams, "Improving developer activity metrics with issue tracking annotations," in *Proceedings of the 2010 ICSE Workshop on Emerging Trends in Software Metrics*, ser. WETSoM '10. ACM, 2010, pp. 75–80.

[22] P. V. Singh, "The small-world effect: The influence of macro-level properties of developer collaboration networks on open-source project success," *ACM Trans. Softw. Eng. Methodol.*, vol. 20, no. 2, 2010.

[23] D. Surian, D. Lo, and E.-P. Lim, "Mining collaboration patterns from a large developer network," in *Reverse Engineering (WCRE), 2010 17th Working Conference on*, 2010, pp. 269–273.

[24] J. Xu, Y. Gao, S. Christley, and G. Madey, "A topological analysis of the open source software development community," in *Proceedings of the 38th Annual Hawaii International Conference on System Sciences*. IEEE Computer Society, 2005, pp. 198.1–.

[25] A. Bacchelli, T. Dal Sasso, M. D'Ambros, and M. Lanza, "Content classification of development emails," in *Proceedings of the 2012 International Conference on Software Engineering*. IEEE Press, 2012, pp. 375–385.

[26] P. Wagstrom, J. Herbsleb, and K. Carley, "A social network approach to free/open source software simulation," in *Proceedings of the 1st International Conference on Open Source Systems*, Genova, Italy, 2005.

[27] E. Shihab, Z. M. Jiang, and A. E. Hassan, "On the use of internet relay chat (IRC) meetings by developers of the GNOME GTK+ project," in *Proceedings of the 2009 6th IEEE International Working Conference on Mining Software Repositories*. IEEE Computer Society, 2009, pp. 107–110.

[28] M. S. Elliott and W. Scacchi, "Free software developers as an occupational community: Resolving conflicts and fostering collaboration," in *Proceedings of the 2003 International ACM SIGGROUP Conference on Supporting Group Work*, ser. GROUP '03. ACM, 2003, pp. 21–30.

[29] T. D. LaToza, G. Venolia, and R. DeLine, "Maintaining mental models: A study of developer work habits," in *Proceedings of the 28th International Conference on Software Engineering*, ser. ICSE '06. ACM, 2006, pp. 492–501.

[30] C. Haythornthwaite, "The strength and the impact of new media," in *Proceedings of the 34th Annual Hawaii International Conference on System Sciences (HICSS-34)-Volume 1 - Volume 1*. IEEE Computer Society, 2001, pp. 1019–.

[31] K. Crowston and J. Howison, "The social structure of free and open source software development," *First Monday*, vol. 10, no. 2, 2005.

[32] M. Zhou and A. Mockus, "Does the initial environment impact the future of developers?" in *Proceedings of the 33rd International Conference on Software Engineering, ICSE 2011, Waikiki, Honolulu, HI, USA, May 21-28, 2011*. ACM, 2011, pp. 271–280.

[33] A. Begel, Y. P. Khoo, and T. Zimmermann, "Codebook: discovering and exploiting relationships in software repositories," in *Proceedings of the 32nd ACM/IEEE International Conference on Software Engineering*. ACM, pp. 125–134.

[34] K. Ehrlich and N. S. Shami, "Microblogging inside and outside the workplace," in *ICWSM*. AAAI Press, 2010.

[35] K. Dullemond, B. v. Gameren, M.-A. Storey, and A. v. Deursen, "Fixing the "out of sight out of mind" problem: One year of mood-based microblogging in a distributed software team," in *Proceedings of the 10th Working Conference on Mining Software Repositories*, ser. MSR '13. IEEE Press, 2013, pp. 267–276.

[36] J. Zhang, Y. Qu, J. Cody, and Y. Wu, "A case study of micro-blogging in the enterprise: Use, value, and related issues," in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. ACM, 2010, pp. 123–132.

[37] D. Zhao and M. B. Rosson, "How and why people twitter: The role that micro-blogging plays in informal communication at work," in *Proceedings of the ACM 2009 International Conference on Supporting Group Work*. ACM, 2009, pp. 243–252.