LINCHPIN DEVELOPERS IN OPEN SOURCE SOFTWARE PROJECTS

Erika Ingrid Mastranzo Corona
Free University of Bozen-Bolzano
39100 Bolzano, Italy
email: erikai.mc@gmail.com

Bruno Rossi
Free University of Bozen-Bolzano
39100 Bolzano, Italy
email: bruno.rossi@unibz.it

ABSTRACT
In Open Source Software (OSS) development, the so-called linchpin developers are those that contribute contemporaneously to several projects, contributing to keeping the community tied together. While such developers have been identified in previous research, their importance within the OSS community has not been widely discussed. The main objective of this work is to analyze their ”weaving” role across projects. With this aim, we mined software repositories, using text mining techniques as log-likelihood ratio and co-word analysis, further building social networks of developers within emerging communities.

The findings show that linchpin developers generally attach in a preferential way to projects in a single specific domain. They tend to be more “project managers”, and “all-hands-persons”, meaning that they bring multi-disciplinary experience across projects. They tend to cover the same role across projects. They generally have high centrality in their projects, and contribute to create projects that ease the transition from a fragmented projects community to a more core-periphery community.

KEY WORDS

1 Introduction
The analysis of the Open Source Software (OSS) community is relevant to evaluate the process of software development, and how actors collaborate and integrate practices for process improvement. Through the study and understanding of the community we can identify new patterns of collaboration, patterns that have influenced the success of projects and at the same time the identification of patterns that produced failures.

The main objective of this work is the analysis and understanding of a specific kind of developers, identified by previous researchers as linchpin developers. These are sort of gatekeepers among different projects, letting knowledge flow across them.

Various definitions of linchpin developers have been given. Madey et al. define them as “...actors who play roles in bridging disparate groups into one large cluster by being the only joint participant in two different projects” [9].

While Xu et al. reported that “...linchpin developer is a member who joins to multiple projects these members link projects into clusters” [13]. On the other hand, Krebs et al. discuss network weavers, according to their definition, “the weaver has the vision, the energy, and the social skills to connect to diverse individuals and groups and start information flowing to and from them these members bring in and take out information” [8].

In characterizing the community building process, we define two goals. The first goal is to find evidence about the domains and roles covered by linchpin developers, characterizing them in comparison to other developers. The second goal is to characterize their relevance in linking together several projects by building an OSS community.

The paper is structured as follows. In section 2, we discuss background, and previous works. In section 3, we delve into the research methodology adopted, giving details about the techniques used for the subsequent analysis part. Section 4 provides descriptive statistics about the context of linchpin developers sampled, and sections 5,6,7 are the core part of analysis, with details on linchpins domains, roles, community building and centrality. Sections 8 and 9 present discussion, and conclusions.

2 Background & Previous Works
A linchpin developer is a member of the OSS community. Many authors mention the importance of these developers that collaborate simultaneously to different projects within the OSS community. Linchpin developers are actors that play the role to bridge disparate groups into one large cluster. They are joint participants in two or more projects [9].

According to Xu et al., a linchpin developer is in fact a member of a development team that joins multiple projects: these members link projects into clusters, very often participating with different roles.

As such, the identification of roles within the OSS community acquires greater importance. Xu et al. determined that core developers and co-developers have a crucial role connecting the OSS community. Through them, the communication flows faster. They concluded that large projects are composed mainly by co-developers and active users, while small projects are formed by project leaders and co-developers [14]. Madey et al., reported that the open source movement is not a random graph where new nodes are attached to existing nodes with uniform probabilities.
Concas et al. found out that the most active members have more contact with the community, a sort of linchpins between developers and users. Peer support is given when members ask for explanations or support. The collaborative learning is provided when members share their knowledge [2].

In this paper, to model the participation of linchpin in the community, we use the phases that describe the weaving process as described in [8]. We report the phases in Fig. 1. The process of weaving starts at the phase called scattered fragments, in which there are several unconnected clusters. The role of network weavers is to join the different groups or clusters (phase hub-and-spoke), and then more weavers come to provide better connection across groups (phase multi-hub small world network). Afterwards a whole core-periphery structure emerges.

If we consider an OSS community and the fragments different project, we can see that reaching a full core-periphery structure would require a set of developers maintaining the core part across projects.

3 Methodology

The research methodology followed can be categorized as quantitative observational research. We mined software repositories deriving quantitative indicators built upon various techniques to reach the research goals. We used empirical methods commonly used within the OSS research area [12].

From a procedural point of view, we gathered information from several software repositories, and then data has been integrated for the analysis. Gathering and analyzing information about linchpin developers means crossing information about several projects. The datasources used have been FLOSSmole [5], providing statistical data about the domains of the projects and all the roles that have been reported when registering the projects. We associate domain, and roles "tags" to developers: these techniques are appropriate to find relationships among terms, a common problem in which text-mining techniques excel.

We use SNA techniques as we analyze the community building process: we use betweenness centrality the measure of the centrality of the authors within the projects.

3.1 Log-Likelihood Ratio

If we compare the linchpins to other developers, from the point of view of their roles and domains, we can build two separate text corpora with all the terms used for their definition. If we build the distribution of the terms in the two cases, we want to know how the distribution of terms is different, and whether such differences are significant. To perform this process, we use the log-likelihood statistic, a technique that allows to get a ratio that tells us the more different terms - according to the relative occurrence - between two texts [7].

The process of calculation is the following. Once we have the frequency list for both the texts, we need to create a contingency table, as in Table 1. This table is calculated for every term in the union of the two text corpora sets. In the table, $a$ and $b$ are the frequencies of the same term in the two texts, while $c$ and $d$ represent the total of term in the two texts. We can think about $a$ and $b$ as the observed values ($O$) in the next formula for the calculation of the expected values ($E$):

$$E = \frac{(a + c)(a + d)}{N} \cdot \frac{(b + c)(b + d)}{N}$$

where $N$ is the total number of observations.
We determine the phases in the weaving process by identifying the first linchpin that joins or creates a second project within the community of projects. Once this happens, we consider the community as an \textit{hub-and-spoke} network according to [8]. Once a second linchpin developer joins, the community becomes a multi-hub small world network. As we identified the linchpin developers, we consider how central linchpin developers are within their projects. The concept of centrality in a network is based on the concept of distance among vertices. We consider the distance as the number of steps that are necessary to reach one node within the network. The average over all the nodes, constitutes the average distance. Betweenness centrality gives us the central position of a the developer within the network.

We built the networks for the calculation of centrality by collecting information about all the files within a project, and all the commits of developers within the same network. This took us to a 2-mode network in which every developer is associated to one or more files within the project, if and only if the developer made a code commit in the context of the file. Usage of files for building the collaboration network is common in research (e.g. [1]). Such 2-mode network has then been transformed in an undirected graph, in which two developers are connected by an edge if they worked on the same file. The strength of the edges depends on the number of files in common.

### 4 Descriptive Statistics

Linchpin developers are a minority in the overall OSS community, compared to developers that participate to a single project. This has been already reported in literature, together with the skewness of the distribution of developers by number of projects they are associated with [9]. We compared our dataset, mined from the FLOSSmole repository [5], based on Sourceforge data, with the one given in [9]. Also in our case, the distribution is similarly skewed. Note, though, that we cannot fully reject the null hypothesis that there is no difference between the two datasets, a two-tailed Mann Whitney U test at the 0.05 level of significance, reports $U = 149.0, p = 0.019 (p = 0.019 < 0.05)$. The same can be noted from other major repositories. We compare Sourceforge, Rubyforge and the repository of Free Software Foundation (FSF), according to the minimum, maximum, average, and standard deviation of the number of projects per developer (Table 2).

In the remaining part of the paper, to differentiate among types of linchpins, we refer to linchpin developers according to the number of projects they have been active on. We denote them as $L_n$, where $n$ identifies the number of projects. $L_2$ denotes as such linchpins active in 2 projects. We refer in the following to $n$ as the class of the linchpin, and we focus on the Sourceforge repository.
To analyze the association of each linchpin developer with different software domains, we gathered each of the domains relative to the projects they have been working on. In this way, we reconstructed a tuple \( \{ L_n, \{ d_1, \ldots, d_n \} \} \), in which the number of domains for each linchpin is dependent upon the \( n \), the number of projects of the linchpin. At this point we have two different texts that have the association of the terms with the id of the developer. On one side we have all the terms for single-project developers, on the other side, those for linchpin developers. From the distributions, we extracted the most frequent terms according to the domains and the class of developers.3

“Software development” is the most frequent term, raging from \( L_2 \) (5.4% of the total) to \( L_{12} \) (11.74%). It is followed by “Dynamic Content” (2.80%-4.58%), “Site Management”, “Games/Entertainment”, “Frameworks” follow, with typical percentages around 2%-3%. By plotting the evolution of the frequencies, we can derive that there are no major differences in percentages among classes of linchpin developers. Distribution of major domains is almost constant whether developers work in 2 or more projects.

5 Linchpin Domains

In the Sourceforge repository, each developer is associated to a software project. When the software project is created, the creator assigns it to an application domain. Typical cases are “Software Development”, “Games/Entertainment”, “Multimedia”, etc...

To capture: a developer working in the same domain in different projects can be at the same time in one domain within two projects. As such, the relationship would be “Software Development” domain in one project and “multimedia” domain in another one. As such, the relationship would be “Software Development” domain in one project and “multimedia” domain in another one. We report a ‘+’ if the specific term appears more in the context of the linchpin developers, conversely a ‘-’ indicates more relevance of the term for single-project developers. For example, the “Role-Playing” domain is a term with a relative frequency that differs the most between single-project developers and linchpins, the ‘-’ symbol indicates that is more common among single-project developers. We keep the same symbols also for more similar terms, although in that case the meaning is limited, as the relative frequencies are similar.

We also report a ‘+’ if the specific term appears more in the context of linchpin developers and ‘-’ if the term is more relevant for single-developers. Our interpretation, is that some of these domains are more appropriate to be developed by a single developer (e.g. “Turn Based Strategy”), than tools such as “Compilers” that require joint effort among different developers.

5.2 Domains Co-Word Analysis

We run co-word analysis on our classification of linchpins. This analysis represents the preference of linchpin developers to work on given domain(s), given the other choice made. We can exemplify: we can have a linchpin developer that is working in the “software development” domain in one project and “multimedia” domain in another one. As such, the relationship would be “Software Development” domain in one project and “Multimedia” domain in another one. This is a dynamic of domain participation that we want to capture: a developer working in the same domain in different projects

We noted that “Software Development” is the most relevant term in self-relationship. Such pattern is not only for this term, it is more likely that a linchpin that works in a...
domain, will work in the same domain in other projects. If we ignore self-relationships, which are the most important combinations of terms? The main results by analyzing the co-citation maps are the following, in decreasing order of association ($S$):

1. Site Management ↔ Dynamic Content ($S=0.035$);
2. Build Tools ↔ Software Development ($S=0.016$);
3. Software Development ↔ Dynamic Content ($S=0.011$);
4. Frameworks ↔ Software Development ($S=0.011$);
5. Software Development ↔ Code Generators; ($S=0.10$);
6. Games/Entertainment ↔ Role-Playing ($S=0.009$);
7. Site Management ↔ Software Development ($S=0.005$);
lowing combinations (with association index $S$):

1. Project Manager ↔ Developer ($S=0.189$);
2. All-HandsPerson ↔ Developer ($S=0.060$);
3. All-HandsPerson ↔ ProjectManager ($S=0.021$);
4. Developer ↔ Advisor/Mentor/Consultant ($S=0.019$);
5. Developer ↔ Package(rpm,.debetc) ($S=0.011$);
6. Tester ↔ Developer ($S=0.009$);
7. Developer ↔ UnixAdmin ($S=0.006$);

7 Community Building Process

We investigated the process of the creation of OSS communities, in which linchpins play a major role. We followed the representation of network weavers in [8], together with the representation of project phases to which weavers contribute, from fragmented, to hub and spoke, and weak ties. We wanted to see how linchpin developers are involved in the community building process.

To perform such investigation, we sampled a number of developers\(^6\). Reason is, for this kind of analysis we need to get information about commits from all the developers within projects. Considering the initial sample, this would be infeasible. As a sample selection strategy, we picked randomly 10 linchpin developers out of the overall linchpins population. This meant mining data for 29 projects, and 793 developers overall.

By analyzing the linchpin developers in our sample, we observed two different behaviours when transitioning from fragmented structure, to a hub and spoke structure. There are cases in which linchpins create a new project to which the other developers attach, while in other cases linchpins join already existing projects.

We can observe the behaviour of linchpins in Fig. 2. We see four snapshots of one community that is going to be built in time. To trace the community, we followed the activity of linchpin developers and all the projects they have been involved in.

The figure shows the emergence of a three-projects community. In this community, there are two linchpin developers, jorrit and sunshine, that we evidenced in the figure with different colours. Each developer is represented as a node, and there is an edge between two developers, if and only if during one period they worked on at least on file in common. In Fig. 2.a there is one project within this community. In Fig. 2.b, we show a two-months period in which a new project, CEL, emerges. The creation of such project is the effort of one of the linchpin developers (jorrit). In Fig. 2.c, a second linchpin developer (sunshine) joins the project some months later. In Fig. 2.d, we can see how the development connections intensify in the main and in the second project, but the process of attachment of linchpins to a new project follows the same pattern: developer jorrit creates a new project, to which also sunshine joins\(^7\).

\(^6\)the list is available online

\(^7\)if we would follow the projects’ history from this point, we would see that these developers would be the only ones in this project
The process follows a similar pattern as in [8], in which we start from a fragmented structure of the community, and the network weavers (in our case the linchpins) start the connecting process among different projects. In our sample of 10 developers, linchpins take part to overall 32 projects. In 7 cases, linchpins are the creators of such projects, while in 25 cases they attach to existing projects.

To evaluate the importance of linchpin developers, we analyzed the distribution of the betweenness centrality of linchpin developers compared to the rest of developers within the projects. To ease the comparison, we use beanplots instead of traditional boxplots. A beanplot shows each data point as a small line in the diagram. Near to each line, there is the estimated average distribution density at the specific point, so that it is easier to see where most of the data is clustered [6].

We found that linchpin developers have higher centrality compared to the other projects' developers (Fig. 3). We can reject the null hypothesis that there is no difference between the two datasets, a two-tailed Mann Whitney U test at the 0.05 level of significance, reports \( U = 14,894, \ p = 0.000013 \) (\( p = 0.00013 \ < 0.05 \)) with \( n_1 = 26 \) and \( n_2 = 801 \). So, there is a significant difference among the linchpin developers and other developers in terms of betweenness centrality within projects.

We analyzed further the differences of linchpins in their betweenness centrality across projects. We found that linchpin developers that build software projects have higher centrality compared to linchpin developers that follow those developers (Fig. 4). We can reject the null hypothesis that there is no difference between the two datasets, a two-tailed Mann Whitney U test at the 0.05 level of significance, reports \( U = 87, \ p = 0.023 \) (\( p = 0.023 \ < 0.05 \)) with \( n_1 = 5 \) and \( n_2 = 21 \). So, there is a marginally significant difference among the two different samples of developers in terms of betweenness centrality.

From these distributions, we derive that, in our sample, linchpin developers have a central role in the development process within the project, and that - among them - the linchpins that create new projects, keep a bigger central role within the community.

As such, they do not only have a weaving role in keeping projects tied together, but also a central role within projects.

8 Discussion

The aim of this paper was to characterize linchpin developers in the OSS community building process. We defined two goals. The first goal was to find evidence about the domains and roles covered by linchpin developers, characterizing them in comparison to other developers. The second goal was to characterize their relevance in linking together several projects by building an OSS community.

Linchpin developers tend to cover the same roles across projects. Even though some changes of roles are happening, mostly linchpins maintain their role. Domains are not so strict within the context of linchpins, as roles. Typically, if a linchpin works on a project, he tends to keep the same domain also in other projects, although some more variance is foreseen compared to roles. This can be seen as confirmation of the non-randomness of graph attachment to software projects [9].

Even though linchpin developers work across several projects, they tend to cover central roles within the commits network. This is not an irrelevant implication, as according to [10], “bugs reported by core members are solved more rapidly when more duplicates are found [...]”, and according to [3], core developers perform core development activities, while peripheral developers solve smaller issues.

We found confirmation from the research in [15], that the combination “Software Developer”\( \rightarrow \)“Project Manager”, is the most common within projects. Our angle was a bit different, in that we focused on the developers - getting typical roles combinations within projects, and not on projects - getting the most relevant combinations of roles within projects. This meant that in [15], also single project developers were mined.

In our analysis, we found the importance of “All-hand-persons”: our interpretation is that they bring more multidisciplinary competence to projects. This is certainly positive, considering that according to [16], projects with higher number of roles are those that are more successful (in terms of ranks).

We found confirmation about the fact that core developers join several projects [14], while our analysis could not support the findings in [2], due to the difference in networks that have been analyzed.

Overall, from our exploratory analysis of linchpin developers, we contributed with the following findings:

- there are different types of linchpin developers, according to the weaving process across projects. There are developers that create sub-projects, and there are linchpin developers that attach subsequently. Subprojects are mostly related to the main one;
- in our sample, we found more linchpin developers that follow rather than create new projects;
- compared to other participants in the projects, linchpin have a high centrality within the developers network, this reports about their relevance within the projects;
- giving the fact that they cover a central role, they tend to be mostly developers and project managers;

9 Conclusion

Linchpin developers are sort of project “weavers” that keep different projects tied together. By means of the current research, we found evidence about the importance of their activity.

Linchpin developers are mainly in the central part of the networks. Linchpin developers share knowledge across
projects within the same domain. It has been found a preference of attachment to domains in which linchpins have already experience. As well, the same preferential attachment has been found for roles. There are roles that linchpin developers cover with a certain frequency, such as being developer in one project, and project manager in another one. Our interpretation is that, while they do not bring the same effort in the other project, they share knowledge within the same or similar domains.

We contributed to the research on knowledge spreading within OSS, by finding empirical evidence of the emergence of a process that from a fragmented network, we reach a hub and spoke structure, with linchpin developers playing the major role in this activity. We found that linchpin developers play a central role within their projects, but those that are the builders - and extend the community - are more central within the projects. In this sense, linchpins have a sort of weaving role, by extending the community built typically around one main project. From this exploratory analysis, we can state the importance of linchpin developers in the OSS community. Going one step further, with the future works, we can investigate whether the presence of such developers can be an indication of the healthiness of a software community.

References


