Visualizing Discord Servers

Marco Raglianti, Roberto Minelli, Csaba Nagy, Michele Lanza

REVEAL @ Software Institute – USI, Lugano, Switzerland

***Abstract*—The last decade has seen the rise of global software community platforms, such as Slack, Gitter, and Discord. They allow developers to discuss implementation issues, report bugs, and, in general, interact with one another. Such real-time communication platforms are thus slowly complementing, if not replacing, more traditional communication channels, such as development mailing lists. Apart from simple text messaging and conference calls, they allow the sharing of any type of content, such as videos, images, and source code. This is turning such platforms into precious information sources when it comes to searching for documentation and understanding design and implementation choices. However, the velocity and volatility of the contents shared and discussed on such platforms, combined with their often informal structure, makes it difficult to grasp and differentiate the relevant pieces of information.**

We present a visual analytics approach, supported by a tool named DISCORDANCE, which provides numerous custom views to support the understanding of Discord servers in terms of their structure, contents, and community. We illustrate DISCORDANCE, using as running example the public Pharo development community Discord Server, which counts to date ∼180k messages shared among ∼2,900 developers, spanning 5 years of history. Based on our analyses, we distill and discuss interesting insights and lessons learned.

***Index Terms*—visualization, software communities, Discord**

I. INTRODUCTION

Ever since the advent of internet, digital communities have been born, have thrived, and have also died out. Early platforms were purely text-based (*e.g.,* mailing lists, Internet Relay Chat), while modern platforms, such as Slack[1] and Discord,[2] are full-blown multi-media environments with high velocity and throughput. Global software communities are scattered around the planet and, also driven by the open source movement, have embraced such platforms early on. Each software community uses various communication mechanisms to keep in touch and to discuss [1]. While some of those communication channels can be mined fruitfully [2], [3], the signal-to-noise ratio of certain of those channels is low [4].

In the last decade, more feature-rich alternatives have emerged. Rich content media sharing in instant messaging software (*e.g.,* Slack) broadened the spectrum of possible interactions between members of these virtual communities and turned such platforms into precious information sources. Recently, tools originally targeted at video-gaming communities (*e.g.,* Discord in Fig. 1) have seen an increasing adoption in other contexts, such as classrooms [5] and software developers communities at large.[3]

[1]See `https://slack.com` [accessed 2021/08/06]
[2]See `https://discord.com` [accessed 2021/08/06]
[3]See `https://git.io/JnRGr` [accessed 2021/08/06]

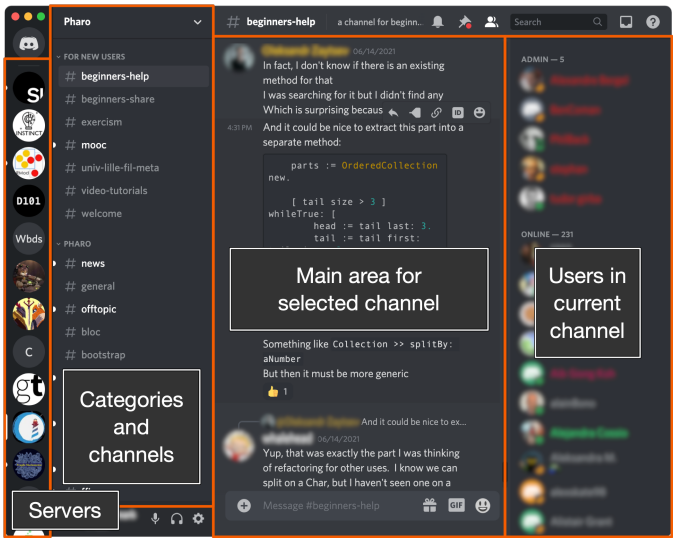

Fig. 1. A Screenshot of the Discord Application

Developers use these communication channels to promote the libraries/frameworks they developed and offer technical support. Novice developers can ask for help and receive answers from their more experienced peers. In a nutshell, these platforms act as a novel source of documentation and encapsulate design decisions and implementation choices. However, as already pointed out by Jaanu *et al.* [4], the velocity, volatility, and transient nature of the information exchanged on such platforms, combined with their informal structure, makes it difficult to grasp and differentiate the relevant pieces of information.

We present a visual analytics approach, supported by a tool named DISCORDANCE, which provides a catalogue of custom views to support the understanding of Discord servers in terms of their structure, contents, and community.

Our approach aims at easing the comprehension of relevant aspects about the community and its individuals. Apart from the approach and the presentation of DISCORDANCE, the main contribution of this paper is a set of custom views to progressively disclose information about:

1) the ***server***, its structure, and its content subdivision;
2) individual ***channels***, their history, and their potential information content;
3) ***authors*** as individual entities, their different activity patterns, and their interactions with the community;
4) ***source code elements*** that can be mined for insights on domain-specific aspects.

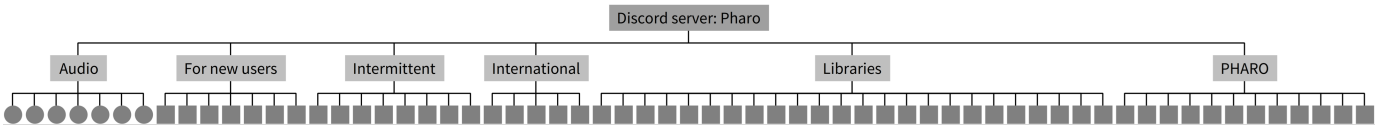

Fig. 2. Structure of the Pharo Discord Server (root): Categories (rectangles), Voice Channels (circles), and Text Channels (squares)

II. Background

A. Discord in a Nutshell

Discord is a *Voice over Internet Protocol* (VoIP), instant messaging, and digital distribution platform. It can be seen as a client/server application with additional support for peer-to-peer communication. A ***Discord Server*** is the basic functional unit encapsulating the concept of a community. A server is typically divided into ***Categories*** and ***Channels*** (Fig. 2). The two main channel types are Text and Voice. Text channels support textual messages, embedded links (*i.e.,* textual messages with a partial preview of the linked resource), emojis, reactions, and file sharing. Voice channels support spoken communication, camera feeds, and screen sharing. To better structure a Server, Channels can be grouped into Categories. Discord uses a ***permission system*** based on ***roles*** assigned to the members to limit (or grant) the visibility of a given channel (or a category) to a given role. Members of a server interact with each other in the channels they have access to. In a server, there are two types of users: Regular (*i.e.,* humans) and Bots (*i.e.,* software applications that run specific activities in a channel, *e.g.,* moderation).

Our supporting tool, called DISCORDANCE, features a bot that can be added to a server to retrieve and analyze its data, *e.g.,* messages it is entitled to read. DISCORDANCE also uses DiscordST[4] [6], a client for the public Discord REST API written in Pharo.[5] After creating a domain model by scraping the Discord server, DISCORDANCE enables its interactive visualization based on the views presented in Section III.

B. Case Study: The Pharo Discord Server

We analyze the Pharo development Discord server. Pharo is a pure object-oriented programming language and a powerful environment, focused on simplicity and immediate feedback, inspired by Smalltalk. Table I provides statistics about this server, containing several hundreds of people with an average of 100 messages per day.

TABLE I
STATISTICS ON PHARO DEVELOPMENT DISCORD SERVER

Snapshot Date	Jun 16 2021
First Message Date	Sep 8 2016
Activity Duration	4 years 282 days
# Active Members	966
# Inactive Members	1,525
# Previously Active Authors	394
# Sent Messages	183,481

[4]See `https://git.io/JnR3h` [accessed 2021/08/06]
[5]See `https://pharo.org/` [accessed 2021/08/06]

III. Visualizing Discord with DiscOrDance

DISCORDANCE offers six polymetric views [7] to analyze different aspects of a Discord server such as channels, authors, and source code elements discussed in messages.

A. Channel Activity View

This view provides an overview of the channels in terms of their activity, *i.e.,* the number of messages sent. Each text channel is represented as a rectangle: The height is proportional to the number of messages sent to that channel while the width is proportional to the number of authors who sent them. The area of rectangles indicates the "activity" in a given text channel. Voice channels are represented as fixed-size circles. Since channel names are not unique, to distinguish them, we keep track of the channel hierarchy (*i.e.,* categories containing them). For this reason, the Channel Activity View adopts a tree layout, with categories at the top.

Examples: Fig. 3 depicts the Channel Activity View for the Pharo Development Discord Server. At a glance, we can spot the two most active text channels: *"general"* in the *"PHARO"* category and *"beginner-help"* in the *"For new users"* category. The former counts 49,872 messages from 862 authors while the latter counts 23,305 messages from 494 authors.

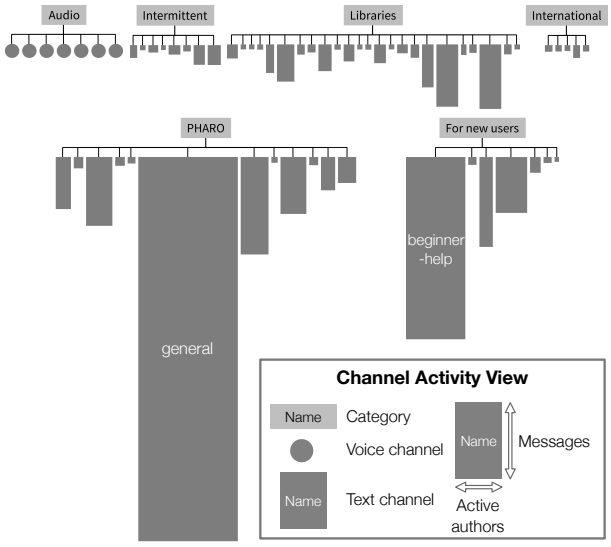

Fig. 3. Channel Activity for the Pharo Development Discord Server

B. Channel Activity Timeline View

Channels can also be analyzed in terms of their recent activity and overall lifespan. When considering the first message sent in a channel as the starting point and the last one as the ending point, we can see interesting patterns.

Examples: The server started as a single channel for a few months. Most channels have recent activity and a long history (Fig. 4).

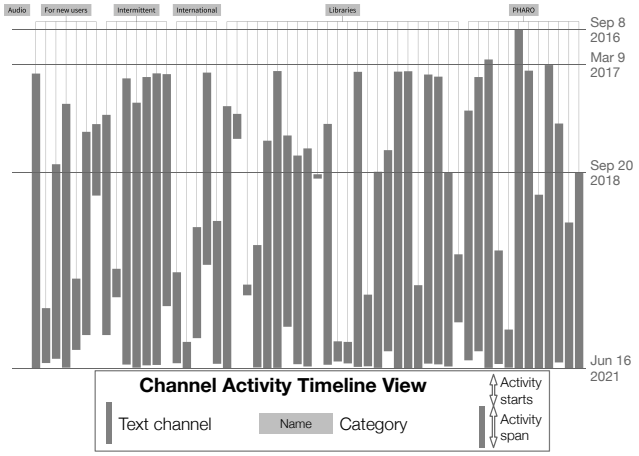

Fig. 4. Channel Activity Timeline for the Pharo Development Discord Server

There are important channels that are still relevant. Their initial activity date and the overall height of the channel's representation can indicate how important the channel has been in the history of the community. In this view, we see channels that were active in the past and are not active anymore and channels created only recently. The former are candidates for archival and probably do not serve any real purpose besides documenting the past. The latter should be investigated separately since their overall activity may be overshadowed by longer standing channels, despite possibly containing interesting insights or patterns.

C. Author Activity Status View

A Discord server is dynamic in terms of members and their activity status. Some authors send a few messages and then quit the community while others only read messages without sending anything. This view aims at highlighting the composition of the user base of a server. Table II summarizes author types and membership status.

TABLE II
AUTHOR TYPES BASED ON ACTIVITY & MEMBERSHIP STATUS

Activity/Membership	Definition
active member	sent messages, currently receiving messages
inactive member	receives messages (and possibly reads them), didn't send any message (yet)
active ex-member	sent messages in the past, not part of the Pharo Discord community anymore
inactive ex-member	never sent any message, presence on the server can be inferred by at least one mention

Examples: Fig. 5 depicts all authors, sorted by decreasing no. of messages, colored by their activity/membership status.

There are 966 active authors who are also current members of the community. 1,525 members did not post a message yet. Previously, 394 authors posted at least one message but they left the server, thus, they are not members anymore.

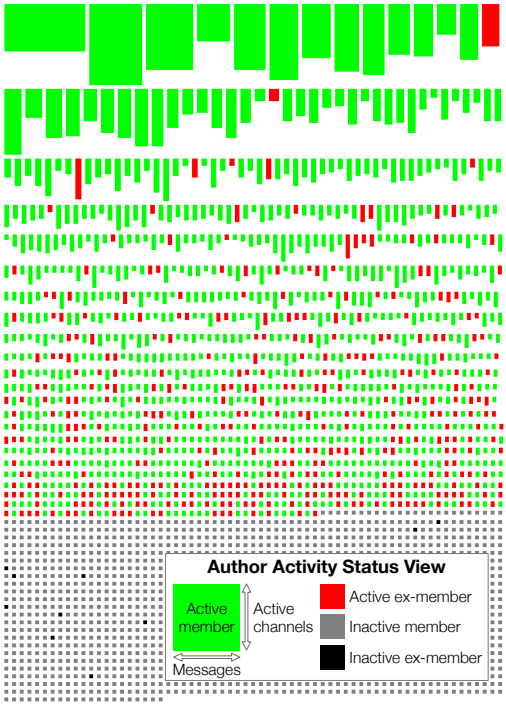

Fig. 5. Author Activity Status for the Pharo Development Discord Server

Moreover, this view highlights differences in author's behavior. For example, Author 1 (*i.e.,* first row, first rectangle) is more active than Author 2 (*i.e.,* first row, second rectangle) but in a significantly lower number of channels.

It would be interesting to investigate the activities of active ex-members to find insights on why they left the community (*e.g.,* they did not get an answer to their first question, they had a flame with another user, they changed topics).

D. Author Activity Sparkline View

This is a chart-based view. Every author can have different activity patterns when using Discord to communicate. Activity charts are a compact representation of daily activity by an author. In the single view (Fig. 6), only one author is considered and his daily number of sent messages can be charted over his activity period or the whole server activity period.

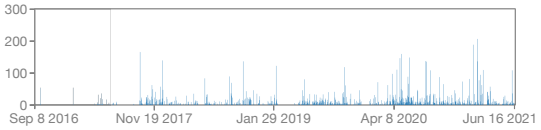

Fig. 6. Author Activity Sparkline for Author 9

Using a small multiples approach [8], we can compare different authors to spot differences in their activity patterns.

Examples: In Fig. 6, we depict the activity of a long-standing member of the community. The increase in average activity in the last year and a half is apparent as well as a certain periodicity in the overall activity that could be further investigated (*e.g.,* with respect to seasonality).

In Fig. 7, we show the top 10 most active authors with their daily activity, charted over the whole server lifetime.

Fig. 7. Author Activity Sparklines for the 10 Most Active Authors

Authors 1, 2, and 3 (from top-left by row) are still active while 4, 5, and 6 have stopped their activity between around 1.5 and 2.5 years ago. Author 7 is the most active, while his activity started more recently compared to the others. Author 10 has a very low average daily activity.

E. Code Blocks View

Many messages feature structured content of various type, such as stack traces and diffs. We are interested in source code, which developers frequently share and discuss using Discord. The Pharo Discord server features close to 14k messages with structured content, and more than 2.3k messages with source code. Discord supports Markdown that allows marking code blocks in a message. We use the syntax highlighting annotation (*i.e.,* ```` ```smalltalk ````) to identify the programming language of a code block, as shown in the sample message below.

````
```smalltalk
MyClass new doThing: (MyClass new doAnotherThing)
```
is equivalent to:
```smalltalk
[ :myClass | myClass doThing: myClass doAnotherThing ]
    value: MyClass new.
```
````

Examples: In Fig. 8, we show all the potential code blocks, using specific colors for those with a recognized syntax highlighting. The vast majority of source code elements are 2,530 Smalltalk code blocks.

F. Class References View

Narrowing down the presence and relevance of source code related information in our case study, this view investigates the number of mentions for specific classes in the Pharo core libraries. We restrict the code blocks to the ones explicitly marked for Smalltalk syntax highlighting. We then perform a regular expression based pattern matching with class names to extract class mentions.

Examples: In Fig. 9, we show mentions of the Collection class hierarchy. We sort them by the number of mentions, thus highlighting the most common classes in Smalltalk code blocks. There are 294 references for the *String* class, 212 for the *Dictionary* class, 185 for the *OrderedCollection* class, etc.

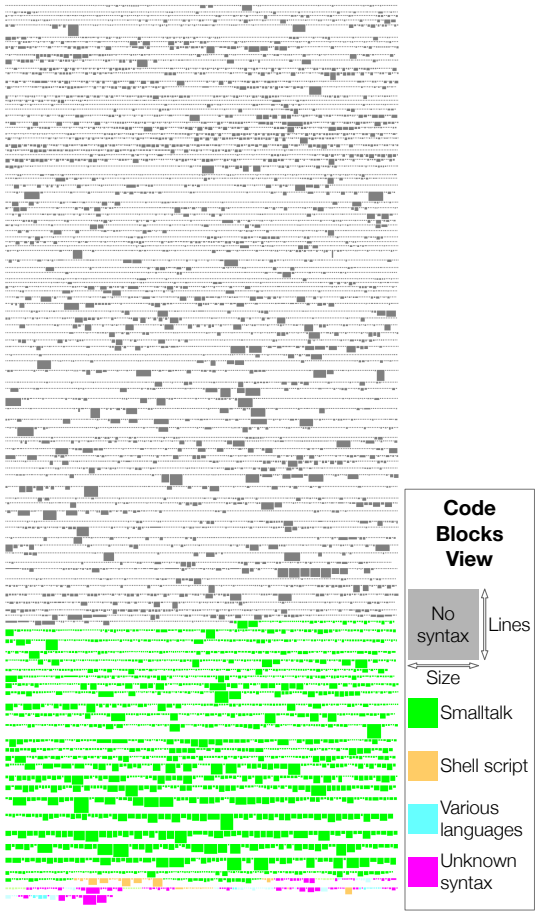

Fig. 8. Code Blocks for the Pharo Development Discord Server

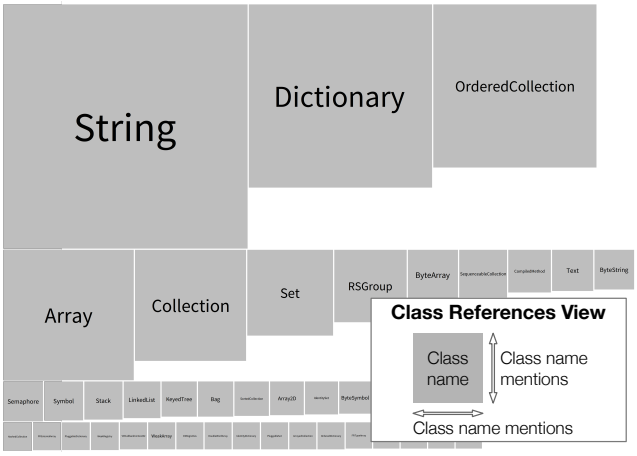

Fig. 9. Class References for the Smalltalk Collection Hierarchy

IV. Related Work

Numerous works have highlighted the importance of communication channels and platforms for a more comprehensive understanding of software systems and their developer communities, *e.g.,* [9]–[16]. The software visualization community has so far neglected them, with some notable exceptions.

Software developer communities visualization has been presented by Stephany *et al.* [17], who showed an analysis and visualization of mailing lists and source code repositories of three open source projects. The authors highlight the underlying social structure and communication patterns between developers within each project. Neu *et al.* presented a contributor-centric visualization in the form of "Developer Activity Diagrams" [18]. Git repositories are mined to extract contributors' daily activities in terms of *git commits* at different scales (*e.g.,* project, ecosystem). Source code, mailing lists, and bug trackers are also mined by Goeminne and Mens [19]. Their integration of different data sources to feed the visualization layer is another step in confirming the importance of communication between developers and between developers and users. Issue trackers, code repositories, wikis, and analytics platforms are possible knowledge sources considered for visualization by Johanssen *et al.* [20].

V. Conclusions & Future Work

The importance of software community platforms is increasing and is fundamentally changing how developers discuss and interact with each other. We presented a set of views, generated using a custom-built tool named DiscOrDance, exploring one instance of such a platform: Discord. Beyond the views that we presented, and the insights that said views allow, our primary contribution is a comprehensive approach, preceded by a careful modeling of the domain, to visually navigate and explore novel types of information, that ultimately all relate to software systems, and their understanding from the point of view of the people developing and discussing them.

As part of our future work, we could compare different communities to extract commonalities and differences as highlighted by the presented views. We could also add new views by exposing features extracted from the messages (*e.g.,* @mentions between authors). Another interesting direction is to use the gained insights to automatically recommend Discord server refactoring (*e.g.,* channel split/merge) or to perform sanity checks (*e.g.,* roles, permissions). Finally, a web-based front-end for DiscOrDance could help communities to explore their own servers and gain actionable insights.

Acknowledgments

We gratefully acknowledge the financial support of the Swiss National Science Foundation (SNSF) for the project "PROBE" (Project No. 172799) and the Fonds de la Recherche Scientifique (F.R.S.-FNRS) and the SNSF for the joint Lead Agency project "INSTINCT." We also thank the Pharo Discord community, and in particular Stéphane Ducasse and Marcus Denker, for adding the DiscOrDance bot to their server.

References

[1] J. A. Teixeira and H. Karsten, "Managing to release early, often and on time in the OpenStack software ecosystem," *Journal of Internet Services and Applications*, vol. 10, no. 1, p. 7, 2019.

[2] A. Bacchelli, T. Dal Sasso, M. D'Ambros, and M. Lanza, "Content classification of development emails," in *Proceedings of ICSE 2012 (34th International Conference on Software Engineering)*. IEEE, 2012, pp. 375–385.

[3] A. Guzzi, A. Bacchelli, M. Lanza, M. Pinzger, and A. Van Deursen, "Communication in open source software development mailing lists," in *Proceedings of MSR 2013 (10th Working Conference on Mining Software Repositories)*. IEEE, 2013, pp. 277–286.

[4] T. Jaanu, M. Paasivaara, and C. Lassenius, "Near-synchronicity and distance: Instant messaging as a medium for global software engineering," in *Proceedings of GSE 2012 (7th International Conference on Global Software Engineering)*. IEEE, 2012, pp. 149–153.

[5] R. Menzies and M. Zarb, "Professional communication tools in higher education: A case study in implementing Slack in the curriculum," in *Proceedings of FIE 2020 (Frontiers in Education Conference)*. IEEE, 2020, pp. 1–8.

[6] J. Cerezo, J. Kubelka, R. Robbes, and A. Bergel, "Building an expert recommender chatbot," in *Proceedings of BotSE 2019 (1st International Workshop on Bots in Software Engineering)*. IEEE/ACM, 2019, pp. 59–63.

[7] M. Lanza, "Codecrawler — polymetric views in action," in *Proceedings of ASE 2004 (19th International Conference on Automated Software Engineering)*. IEEE CS Press, 2004, pp. 394–395.

[8] E. Tufte, *Envisioning Information*. Graphics Press, 1990.

[9] C. M. Costa Silva, "Reusing software engineering knowledge from developer communication," in *Proceedings of ESEC/FSE (28th European Software Engineering Conference and Symposium on the Foundations of Software Engineering*. ACM, 2020, pp. 1682–1685.

[10] R. Abreu and R. Premraj, "How developer communication frequency relates to bug introducing changes," in *Proceedings of IWPSE-EVOL 2009 (ERCIM Workshop on Software Evolution and International Workshop on Principles of Software Evolution)*. ACM, 2009, pp. 153–158.

[11] D. M. German, B. Adams, and A. E. Hassan, "The evolution of the R software ecosystem," in *Proceedings of CSMR 2013 (17th European Conference on Software Maintenance and Reengineering)*. IEEE, 2013, pp. 243–252.

[12] G. Poo-Caamaño, L. Singer, E. Knauss, and D. M. German, "Herding cats: A case study of release management in an open collaboration ecosystem," in *Open Source Systems: Integrating Communities*. Springer International Publishing, 2016, pp. 147–162.

[13] G. Poo-Caamaño, E. Knauss, L. Singer, and D. M. German, "Herding cats in a FOSS ecosystem: a tale of communication and coordination for release management," *Journal of Internet Services and Applications*, vol. 8, no. 1, pp. 1–24, 2017.

[14] P. Mutton, "Inferring and visualizing social networks on internet relay chat," in *Proceedings of IV 2004 (8th International Conference on Information Visualisation)*. IEEE Computer Society, 2004, pp. 35–43.

[15] 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 MSR 2009 (6th IEEE International Working Conference on Mining Software Repositories)*. IEEE, 2009, pp. 107–110.

[16] A. Foundjem and B. Adams, "Release synchronization in software ecosystems," *Empirical Software Engineering*, vol. 26, no. 3, p. 34, 2021.

[17] F. Stephany, T. Mens, and T. Gîrba, "Maispion: A tool for analysing and visualising open source software developer communities," in *Proceedings of IWST 2009 (International Workshop on Smalltalk Technologies)*. ACM, 2009, pp. 50–57.

[18] S. Neu, M. Lanza, L. Hattori, and M. D'Ambros, "Telling stories about GNOME with Complicity," in *Proceedings of VISSOFT 2011 (6th International Workshop on Visualizing Software for Understanding and Analysis)*. IEEE, 2011, pp. 1–8.

[19] M. Goeminne and T. Mens, "A framework for analysing and visualising open source software ecosystems," in *Proceedings of IWPSE-EVOL 2010 (ERCIM Workshop on Software Evolution and International Workshop on Principles of Software Evolution)*. ACM, 2010, pp. 42–47.

[20] J. O. Johanssen, A. Kleebaum, B. Bruegge, and B. Paech, "Towards the visualization of usage and decision knowledge in continuous software engineering," in *Proceedings of VISSOFT 2017 (Working Conference on Software Visualization)*. IEEE, 2017, pp. 104–108.