Navigating New Networks: The AI Revolution in Open Source Software Communities

Rutger Zoetelief

A thesis presented for the master's degree in Information Management

Tilburg School of Economics and Management
Department Information Systems and Operation Management (ISOM)

Supervisor: Dr. P.K. Medappa

July 2024

Email
R.J.R.Zoetelief@tilburguniversity.edu
Student number
2103151

Committee
Supervisor: Dr. P.K. Medappa
The Second Reader: Dr. I.F. Kanellopoulos

Location
Tilburg University
School of Economics and Management
Department Information Systems and Operation Management (ISOM)
Tilburg, The Netherlands

Date
June 6, 2024

Acknowledgements

I present to you my master's thesis "Navigating New Networks: The AI revolution in open source software communities." It was written for the completion of the MSc Information Management at Tilburg University. I was engaged in researching and writing this thesis from January to June 2024.

I would like to express my sincere gratitude to my supervisor Dr. P.K. Medappa, for his knowledge and support. But also introducing me to this really interesting world of open source software and its communities. And I would also like to thank my family and friends, who supported me throughout the completion of my master's.

Abstract

This study explores the impact of Artificial Intelligence (AI) integration on social network dynamics and collaboration patterns within open source software (OSS) communities. The following research question was developed: *"How does the integration of Artificial Intelligence in Open Source Software projects impact the social network dynamics and collaboration patterns, compared to traditional OSS projects?"* Data were collected from GitHub and Google, focusing on pull request activity within repositories. This research adopted the widely accepted two-way fixed effects model (TWFE) to quantitatively evaluate the impact of AI Copilot on OSS projects by comparing changes in outcomes through a difference-in-difference approach. The findings suggest that the hypotheses regarding AI integration in OSS are not supported. However, the thesis provides a preliminary overview of AI's impact on OSS communities and offers methods for constructing social networks based on GitHub data. The study also emphasized the potential drawbacks of using a single data type to define developer relationships, as well as the potential for a more complex or subtle effect of CoPilot implementation. Additionally, it discusses how the usage of averaging techniques for certain SNA metrics could potentially hide significant variations. In conclusion, this study opens up new avenues for future studies, aimed at enhancing our knowledge of the influence on AI-driven OSS projects.

Keywords: Social Network Analysis, Artificial Intelligence, Open Source Software Communities, GitHub Copilot

Contents

List of Figures

List of Tables

1. Introduction

Over the last several decades, businesses and individuals have embraced the use of open source software (OSS) for commercial purposes have contributed to its development (McClean, 2021). When developers and startups have access to existing open source frameworks, AI tools and libraries, they can avoid the need to build everything from scratch. Therefore, it could accelerate the development process as individuals and organizations can leverage the collective efforts of the open source community, which contributed code, algorithms, and solutions.

A report of Red Hat (2022) mentions the significance of enterprise open source, where 95% of its IT leaders' respondents say that it is of huge importance to their overall organization's enterprise infrastructure. A significant portion of modern computing is built upon open source; from known operating systems like Linux to programming languages as Python (McClean, 2021).

The environment of OSS has evolved rapidly with the integration of Artificial intelligence, GitHub (2022) for example, announced the general availability of GitHub Copilot Chat in June 2022. These developments give new challenges and opportunities in software development and project management. Traditional OSS projects have been explored using various methodologies to understand collaboration and community dynamics. For example, Korchar et al. (2021) investigates the transition from closed to open source projects, focusing on community collaboration through interviews and surveys. But this approach can be response biased and might not be effective for larger communities. On the other hand, mining software repositories offers insights into technical aspects like code changes and bug reports but may overlook the social interactions within the community.

Social Network Analysis (SNA) however provides a unique and holistic lens through which the structure and dynamics of OSS projects can be understood, making it a very effective way of studying collaboration patterns and community dynamics. For instance, the study of Schreiber (2023) provides insights into how people and groups interact and collaborate in TensorFlow, an open source software project. SNA captures the complexity of relationships among contributors, can identify key players within a project and it will allow the comparison of network structures across different projects. These advantages align with the subject that is addressed in this study.

The integration of AI into OSS projects introduces a paradigm shift, necessitating a reevaluation of existing SNA frameworks. The complexity of AI-driven development, characterized by advanced algorithms and data-intensive processes, may alter social network structures within OSS communities. This shift raises questions about the applicability of traditional SNA methods and metrics, as observed in OSS projects without AI components.

There is yet a noticeable gap in the literature regarding a comprehensive comparison of SNA applications in traditional OSS projects versus AI-powered OSS projects. While studies those conducted by Wu et al. (2023) explore the role of social and technical dependencies on OSS project success, an examination of these dynamic shifts in AI-driven projects remains unexplored. This gap

shows an opportunity to learn more about the field by analyzing and comparing the social networks of traditional and AI-powered OSS projects. Such an exploration is crucial for understanding the evolving landscape of OSS development, informing strategies for effective project management, community engagement and successful collaboration for open source community platforms.

1.1. Problem Statement

The problem lies in determining how AI integration affects social network structures and collaboration patterns within OSS projects. Although SNA has examined traditional OSS projects, the specific effects of AI on these networks remain less understood. At the same time, not much research has been done on this particular theme. This knowledge gap would therefore present an interesting research topic to explore. The problem is relevant to OSS project managers, developers and contributors who are incorporating AI into their projects. It also concerns academic researchers and organizations relying on OSS for their operations or product development. The integration of AI in OSS projects will help stakeholders better comprehend the social dynamics at play. The following research question has been developed based on the research framework:

Research Question: "How does the integration of Artificial Intelligence in Open Source Software projects impact the social network dynamics and collaboration patterns, compared to traditional OSS projects?"

2. Theory

2.1. Open Source Software Ecosystem

Software that has source code available for anyone to see, edit and improve is known as open source software (OSS). OSS is present on internet based communities where software developers voluntarily collaborate in order to develop software that is needed (Von Krogh, 2003). Open Source helps increase the pace of innovation, leading to free exchange of novel ideas within these communities (Rathee, 2022). While OSS is likewise developed as proprietary software (e.g., MySQL), much is developed by organizationally and geographically distributed teams of developers, in what is described as community-based development by Lee and Cole (2003). Furthermore, OSS is perceived as unrestricted access to source code, in contrast to the commercial world's more traditional closed proprietary software approach (Bonaccorsi, 2003).

The software economy now uses the word "ecosystem" as a common perspective (Messerschmitt, 2003). A software ecosystem is also defined by Manikas and Hansen (2013) as "the interaction of a set of actors with a common technological platform used by several solutions." This ecosystem frequently depends on a common platform that several parties overlay with their own software (Bosch, 2009). OSS ecosystems develop from self-organized and dynamic processes where businesses and volunteers work together in their contribution to software products (Gerber, 2010; Madey, 2002). A platform is used for establishing such an ecosystem. Studies by Kilamo et al. (2012) and Jansen et al. (2009) mention the facets of such a platform. From an engineering standpoint, a software ecosystem offers development process, environment for the entire software project infrastructure, and technology for implementation. Furthermore, in addition to the technical aspects, social, legal and business aspects must also be considered for the ecosystem.

GitHub serves as a platform for hosting collaborative coding projects. It employs a "fork & pull" approach, where developers generate a personal copy of a repository and propose a pull request for the project maintainer to merge their changes into the primary branch. Beyond hosting code, GitHub facilitates collaborative code review and integrated issue tracking, and it incorporates social networking features (Kalliamvakou, 2014; Tan, 2020). GitHub today also serves as the largest developer community in the world with more than 100 million users (GitHub, 2023). The platform has integrated social features and the availability of metadata through an accessible API, which makes it attractive for software engineering researchers.

The introduction of the social features of GitHub has drawn attention to researchers. For instance, the paper of Zöller et al. (2020) studies the collaboration patterns of OSS projects on GitHub by analyzing the pull request submissions and acceptances of repositories. While Moradi-Jamei et al. (2021) utilizes a large-scale historical dataset of 1.8 million GitHub users and their repository contributions. Whereas Dabbish et al. (2012) concentrated their research on a more qualitative

approach, discovering the importance of transparency for large-scale collaborations and communities on GitHub through a series of in-depth interviews with central and peripheral GitHub users.

OSS thrives as a collaborative community where developers exchange code. These communities often evolve into dynamic software ecosystems, supported by platforms like GitHub. GitHub not only facilitates code sharing and review but also enhances collaboration through its sophisticated fork & pull model and integrated social networking features. This environment enables diverse teams to contribute to OSS projects, significantly influencing the pace of software development.

2.2. Social Network Analysis on Open Source Software Ecosystem

The study of Texeira et al. (2015) employed a mixed method approach using archival data with mining software repositories and Social Network Analysis (SNA). However, while this approach would be in line with the same method of this thesis subject, Teixeira's research question focused more on understanding competition within an open source project, utilizing different SNA metrics than those employed in this study. Horta et al. (2022) discuss a more content-driven approach of analyzing open source communities that is extracted from text or social tags, offering valuable insights. But this would require extensive data collection and preprocessing of textual data, which can be time-consuming and complex. Given the limited timeframe for this research, the focus in this study will be on a more traditional SNA.

Various methods are applied to analyze communities focusing on a broader understanding beyond the more structural analysis of social networks. The role of social comparison theory is emphasized by Lumbard et al. (2024) on understanding how open source community health. Which argues how community dynamics are influenced by social comparisons rather than just network structures. Case studies from Kilamo et al. (2012) and ethnographic studies from Sigfridsson and Sheehan (2011) can all provide information about the social network dynamics and collaboration patterns of these open source communities. However, this paper adopts a data-driven approach rather than a qualitative one for several reasons. SNA allows for analysis of complex relational data; this is especially necessary when these communities have thousands of contributors who may interact with each other. SNA focuses on measurable objectives such as centrality and density which is crucial for assessing the impact of AI integration on OSS. As a result, it makes for an engaging, time-efficient, and overall holistic approach by enabling a comparative analysis and the utilization of preexisting data.

2.2.1. Social Network Analysis

Social network is recognized as a unique research and structural approach within the social and behavioral sciences; distinct because it assumes of the importance of relationships among interacting units or actors (Freeman, 2004; Wasserman, 1994). Broadly speaking, a social network is

constructed from relational data and can be characterized as a collection of social entities, including individuals, groups, and organizations (Oliveira, 2012). Social network theory models these social entities as nodes in a graph, while the connections between them, denoting relationships, are depicted as the graph's edges. Therefore, if two individuals share a relationship, they are directly connected within this graphical model (Madey, 2002), see figure 1. These relationships can either be of personal or professional nature and can range from casual to a closer familiar bond. Besides social relations, edges can also represent flow of information, money, goods, transactions or similarities among others.

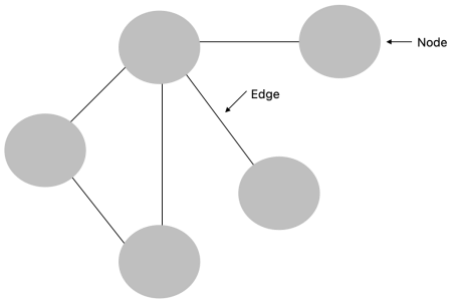


Figure 1: *Social network*

Oliveira and Gama (2012) also discuss how social networks can be modeled using different types of graphs based on the direction of their links. Undirected graphs connect pairs of nodes without any order, meaning the relationship goes both ways equally. Directed graphs, or digraphs, include edges with a set direction, indicating that the relationship between nodes has a specific orientation. A value can also be assigned towards edges where a distinction can be made between unweighted and weighted graphs. Unweighted graphs are either present or not. Whereas weighted graphs provide richer information. According to Granovetter (1973), the weight of a tie is typically determined by its duration, emotional intensity, frequency of interaction, intimacy and service exchange. As a result, weak relationships typically reflect acquaintances while strong ones typically indicate close friends.

2.2.1.1. Network Measures

In this section, we will explore various metrics that are important for analyzing social networks. These measures offer valuable insights into the network's structure, which is crucial for the analysis of this study. Subsequently, these measures can be divided based on the preference to analyze small units, like actors, or the entire network. Both will be considered as they are all relevant to answering the research question.

Actor-Level Measures

Centrality is an over-all measure of how the position of an actor is within the overall structure of the social network and can be computed through to several metrics. In this study, three metrics are chosen (degree centrality, betweenness centrality and closeness centrality), because they are the most frequent used centrality metrics in SNA (Newman, 2010). These measures define the importance of actors within a network. Greater centrality among these nodes or actors, would signify to more

powerful actors in the network since their central position gives them several advantages. For instance, giving them access to other actors more quickly. You can find the definitions of these centrality measures below.

Each of these metrics offers a unique angle on a node's significance. Degree centrality assesses influence established on the number of direct connections a node has, providing a straightforward but somewhat limited perspective since it does not consider the broader network context. On the other hand, closeness and betweenness centrality offer insights based on the global network structure, capturing a node's influence more comprehensively by considering its position relative to all other nodes in the network, thus reflecting its importance from both a local and global standpoint (Sheng J. D., 2020).

- **Degree Centrality:** Measures the number of direct connections a node has (Degenne, 1999).
- **Betweenness Centrality**: Quantifies how often a node acts as a bridge along the shortest path between two other nodes (Oliveira, 2012). High betweenness are often knowledge brokers, connecting multiple communities (He, 2012; McClean, 2021)
- **Closeness Centrality**: Indicates how close a node is to all other nodes in the network, reflecting cohesion or fragmentation (Degenne, 1999).

Network-Level Measures

Network-level measures such as density, clustering coefficient, and average path length provide comprehensive insights into the overall structure and connectivity of a network. These metrics are essential for understanding how densely connected the nodes are, how tightly knit groups form, how efficiently information travels across the network.

- **Density:** Explains the general level of connectedness in a network, representing the proportion of existing links (Degenne, 1999).
- **Clustering Coefficient:** Assesses the degree to which nodes tend to cluster together (Newman, 2010). Considered to be a measure of the “cliqueness” of a network.
- **Average Path Length:** Reflects the overall connectedness and efficiency of information flow across the network. Defined as the average distance between all pair of its nodes (Newman, 2010).

2.2.2. Application of Social Network Analysis on Open Source Software

The literature on OSS through the lens of SNA offer a wide variety of different insights. For instance, Concas et al. (2008) studies successful OSS projects via SNA to analyze developer mailing lists and social network interactions, providing insights into communication flows and coordination within OSS communities. Meanwhile, Torres et al. (2011) applies SNA techniques to understand the role of core groups in fostering community participation. Further, Martínez-Torres et al. (2015) explored the dynamics of OSS communities over time, focusing on the roles of users who join and

leave, and how these changes affect the network's structure. Additionally, Zanetti et al. (2012) provides a detailed analysis of collaboration network in OSS, examining structural features and their implications on community cohesion. Research has also been conducted on open source projects on the platform GitHub utilizing SNA. The study of Kabakus (2020) offers insights into the characteristics of the GitHub network, including popular programming languages, repositories, and developers. Whereas Schreiber (2023) displays the community of TensorFlow through SNA, where they identify key players and relationships between those. Lastly, Wu et al. (2023) examines the role of social dependency networks in developer and module networks.

All studies show significant contribution towards the field. However, this study is one of the first attempts to include the integration of Copilot and its effect it would have on the social network dynamics and collaboration patterns through the lens of SNA. As a result of the fact that Copilot is a very recent AI tool, little research has been conducted on its effect on social network dynamics and collaboration patterns (Friedman, 2021). Thus, this research aims to contribute to the field by exploring this topic. The application of SNA in this research is critical for understanding how the integration of AI might transform these social network dynamics and collaboration patterns in OSS projects. By leveraging SNA, this study aims to uncover how AI technologies influence both the structural properties of networks and behavior of individuals within them, providing a comprehensive view of the changes that the AI integration brings to the environment in OSS. This analysis is crucial for developing strategies that improve the efficiency and cohesion of OSS communities within the AI revolution.

2.3. AI Integration in Open Source Software Ecosystem

The integration of AI into OSS projects is transforming the landscape of software development. AI enables computers to perform complex tasks that mimic human-like cognitive functions, such as natural language processing, decision-making and visual perception (Hazmi Hassri, 2023). This rapid transformation, characterized by the integration of AI into software development practices, is reshaping how code is developed for new software (Zohair, 2018). According to the study of Bird et al. (2022) software development is changing from only writing code to letting the tool write it. Where developers now have to understand the code that the tool writes rather than finishing it themselves. This could indicate a more necessity to comprehend the interactions that take place between developers and AI-powered tools. This adoption is now seen with the use of AI assistants for programming. AI is already used for different goals and in various areas of software engineering (A. Mashkoor, 2022). GitHub Copilot and ChatGPT are examples of these tools that have increasingly been adopted in the past years (Kharrufa, 2023). According to Crawford et al. (2023) AI and ML have the potential to become valuable to software engineering, not just on the active parts of software development, but also on project management side. GitHub Copilot can function similarly to an automatic program synthesis tool. Which is defined as the process where source code is created from

a given definition, such as a description in natural language or through specific examples of what the input and output should be (Sobania, 2022). But fundamentally it is designed as an AI pair programmer that rather assist than completely take over the software development process. Nevertheless, the usage of the technique is already beginning to form a standard tool in software development.

The integration of AI tools like Copilot in OSS projects shows examples of changing effects, serving as a powerful assistant that not only enhances capabilities of developers but also introduces new methods and approaches to software creation. As discussed by Savary-Leblanc et al. (2022) these software assistants empower engineers by enriching their design, construction and maintenance tasks with advanced autonomy and intelligence, fundamentally improving productivity and creativity within the software engineering environment. This large language model is also recognized by Gunnell et al. (2024) as an important trend in scientific computing. Advancements in LLM have helped in the use of application of open source platforms. As the documentation is open and available for many packages. A distinct production version of the model Codex powers GitHub Copilot. The LLM is a GPT-3 based language model with up to 12 billion parameters which has been pre-trained on 159 GB of code samples from 54 million GitHub repositories (Brown, 2020). When using Codex, it shows a good performance in solving a series of handwriting Python programming problems (Chen, 2021). Copilot can be installed as an add-on for the Visual Studio Code development environment (Sobania, 2022). AI-trained tools can possibly move optimization engineers, data scientists and machine learning specialists to higher levels of productivity.

However, some liabilities are discussed in the paper of Dakhel et al. (2023). The authors have examined and compared Copilot code-generation capabilities with humans. The outcomes demonstrate that Copilot is capable of producing accurate and ideal answers for a number of significant algorithm design issues. Nevertheless, the developer's prompt's depth and conciseness could play an important factor in how well-written the resulting code is. Meaning that Copilot can be an asset in software engineering if it is used by expert developers that are familiar with problem context and correct coding improvement methods.

Lastly, Shah (2019) highlight the importance of promoting collaboration between different roles within software engineering, as they can bring diverse expertise to develop holistic solutions. While Wang et al. (2019) describes opportunities for infusing AI into team collaboration practices. Finally, Washizaki (2020) present a vision called "value co-creation of software by AI and developers." The paper addresses the need for collaborative efforts and exchanges between developers, where they should work together with developers and with assistance of AI to achieve this value co-creation. All these papers mention several fundamental transformations within the landscape of software development where collaboration and social network dynamics can play a pivotal role.

2.4. Hypothesis development

Based on the research question and the exploration of relevant theory, hypotheses are developed. The hypotheses are categorized into two subsets: collaboration network and network structure. Each based on the metrics as discussed within the theory. Allowing to capture the social network dynamics and collaboration patterns to address the research question.

2.4.1. Collaboration Network

First of all, hypothesized is that AI integration of Copilot will affect the collaboration network. Various studies emphasize a positive impact on programmers' productivity (Bird, 2022). For instance, the study of Imai (2022) conducted an experiment comparing Copilots programming to that of a developer, where Copilot generated the most lines of code. In another study Dakhel et al. (2023), Copilot demonstrated promising results in its ability to solve most algorithmic problems. These studies indicate an enhancement of productivity for programmers. This could possibly result in more centralized networks on OSS platforms. The study of Schreiber (2023) highlight how productivity in open source software projects often lead to groups becoming more central in project development. Based on that we can argue that AI integration could potentially lead to a higher degree of centrality.

Conversely, AI could also potentially decentralize a community itself by lowering entry barriers for new contributors. Engineers reason, according to Gottlander and Khademi (2023) that widespread of AI coding assistants can lower the profession entry barrier. As a result of the lower entry barrier, new contributors could join and participate in projects more rapidly. The study of Steinmacher et al. (2018) also discuss in how lowering entry barriers can lead to an increase in number of contributors.

Therefore, because of the nuanced view, the following dual outcome is hypothesized:

Hypothesis 1a: *AI integration in OSS will lead to greater centralization of the OSS network.*

AND

Hypothesis 1b: *AI integration in OSS will lead to greater decentralization of the OSS network.*

In graph theory, betweenness centrality represents the degree to which nodes stand between each other, acting as a bridge between two other nodes (Oliveira, 2012). These nodes can link multiple communities (He, 2012; McClean, 2021). Here, theorized is, a decrease in reliance on these specific key developers that form as bridges due to several reasons. AI can lead to the automation of coding task (Dakhel, 2023), and this may reduce dependence on these key developers within a network. Also, because of the tool providing coding suggestions (Savary-Leblanc, 2022), it might decentralize the coding process, resulting in contributors feeling less necessity to ask for advice on

problem solving. Besides the theorization of a reduction in intermediary roles, one could argue that Copilot encourages direct collaboration between team members by providing relevant information through these code suggestions and thereby reducing dependency on knowledge brokers or central nodes for guidance. Hence this conceptualized hypothesis:

Hypothesis 2: *AI integration in OSS will lead to a decrease in the average betweenness centrality of the network.*

Finally, during interactions in the development of a project, users may need help from somebody else but do not have a direct link towards that specific individual. In SNA, closeness centrality can indicate in how many steps a node can reach every other node from a given node (Freeman, 2004); in this case, the person who needs assistance from somebody else. Research like the study of Li et al. (2022) indicate that the introduction of AI can positively impact knowledge sharing. Another research discusses how AI can help redesigning roles and processes, making it easier for knowledge workers (important nodes) to share information (Sundaresan, 2022). Thus, this study hypothesizes a positive relationship between the closeness centrality of a developer network and AI:

Hypothesis 3: *AI integration in OSS will lead to an increase in the average closeness centrality of the network.*

2.4.2. Network Structure

Subsequently, this study also hypnotizes that the introduction of AI affects the network structure of communities on GitHub. The first theorization regards the density, which explains the level of connectedness in a network. In other words, representing the proportion of existing links (Degenne, 1999). According to social network theory, tools that facilitate communication and reduce resistance of interaction may increase network density (Borgatti, 2024). And as mentioned, software engineers speculate that AI potentially will lead to a lower entry barrier (Gottlander, 2023), resulting in an increase of contributors (Steinmacher, 2018). Which could lead to an increase in participation and collaboration, increasing the number of connections within a network. Following up on that, research on AI and organizational connectedness have shown that AI-enabled platforms that enable communication and collaboration have been found to stimulate the overall connectedness of teams, or within this study, networks (Li N. Y., 2022). Therefore, it is hypothesized that AI integration will lead to an increase in density for OSS communities:

Hypothesis 4: *AI integration in OSS will lead to an increase in density.*

Within social network theory, the clustering coefficient is described as the extent to which cliques form within a network (Newman, 2010). As discussed before, AI could potentially enhance knowledge sharing within communities and let individuals work more closely together (Sundaresan, 2022). Combined with the study of Li et al. (2022) that emphasized the role of AI and its effect on organizational connectedness, we theorize that the introduction of Copilot will lead to an increase in the average clustering coefficient. Which means a shift from more fragmented groups within communities towards more cohesive networks within the open source community:

Hypothesis 5: *AI-integration in OSS will lead to an increase in the average clustering coefficient.*

The last hypothesis introduces the average path length. Which describes how fast information can travel from point A to point B within a network. Based on multiple studies, it is argued that this information will spread more quickly than traditional OSS projects. For example, Schreiber (2023) describes how using the AI tool could boost productivity. Furthermore, research by Dakhel et al. (2023) shows that, when utilized by professionals, Copilot can be a useful tool in software projects. Thus, this study argues that appropriate application of AI in software development could result in a more effective and efficient flow of information:

Hypothesis 6: *AI-integration in OSS will lead to a decrease in the average path length.*

2.5. Conceptual model

Based on the argumentation in the previous hypothesis section, we can construct a conceptual model for our research. Additionally, the number of pull requests is incorporated as a control variable, ensuring that the analysis accounts for underlying activity levels per project. In figure 2, the conceptual model is visualized.

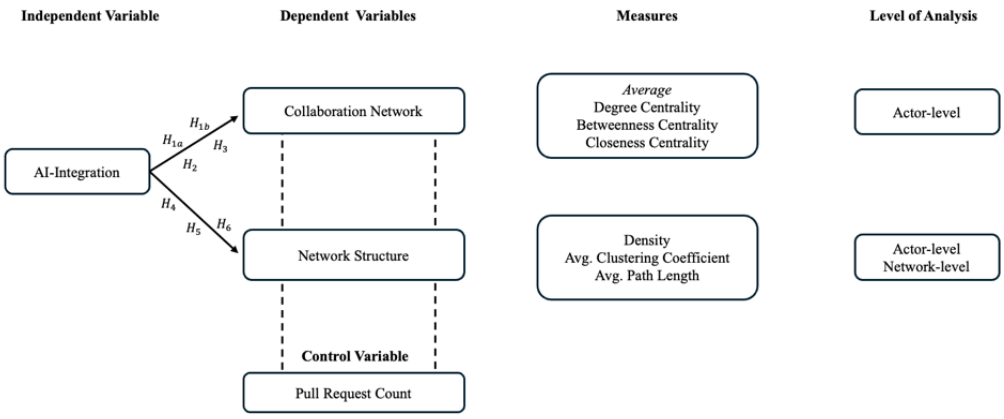


***Figure 2**: Conceptual model*

3. Methodology

The section includes the methodology section, which describes the data collection process used in this study. Here, the development of the construction of the networks is also shown. Following this, the timeframe is discussed here. Additionally, a statistical framework is offered. Lastly, a transparent summary of the use of AI tools in this research is provided in appendix A.

3.1. Data Collection

To empirically test our hypotheses, data were collected from GitHub, using the GitHub GraphQL API, specifically targeting pull request activity within all repositories maintained by GitHub and Google (GitHub, 2001). This data was methodically organized into one panel dataset in April 2024, which includes observations for both companies from 2020 until the end of 2023. Ultimately, a total of 37.9868 observations belonging to 1265 repositories were extracted.

Using the pull request data, the repository networks were constructed. These were aggregated on a monthly basis within the time frame. For GitHub, 2078 networks were aggregated and constructed in total, whereas for Google, 8155 networks were aggregated and built. To elaborate on that, one network corresponds to one repository, and within that repository, a network is built every month if pull request data is available for that specific repository and month.

Based on the constructed networks a dataset was constructed to capture social network metrics for both companies. After the removal of incomplete observations and the outliers' analysis, a more refined dataset was created which allows for a comparative analysis of the technical preview effects of the introduction of GitHub's Copilot. Ultimately, table 1 shows the data that was utilized to capture the difference in pre- and post-treatment for our treatment group GitHub in comparison with our control group Google. Representing the timeframe for the technical preview, which starts from the beginning of 2020 until the public release of July 2022.

Table 1: *Summary of total of social network measures during technical preview effect*

Company	**Social network measures**
GitHub	880
Google	4.168
Total Observations	5.048

The detailed computations of all the social network measures applied in this study, including formulas, are provided in appendix B. The formulas are expressed for an undirected graph and if necessary, the formulas are normalized due to the different sizes of the constructed networks.

3.2. Construction of the Network

After collecting the data from GitHub, the networks were constructed, which is described in this section. In each relational transaction, contributors are identified as nodes. The edges represent the relationships or interactions between these nodes, established through pull requests. Each relational transaction begins with a pull request initiated by a contributor, who is identified as the source node (Oliveira, 2012). The target nodes are those that engage within this pull request through actions such as commenting, reviewing or committing code in response (GitHub, sd). For a link (edge) to be established between two contributors, they must participate in that same transaction, either one being the source and the other a target, or both being targets collaborating within the same transaction.

Figure 3 illustrates the network's construction process. Here, two pull requests are joined based on the same repository ID, year, and month. Appendix C provides a more detailed case study of one of the constructed networks from the company GitHub.

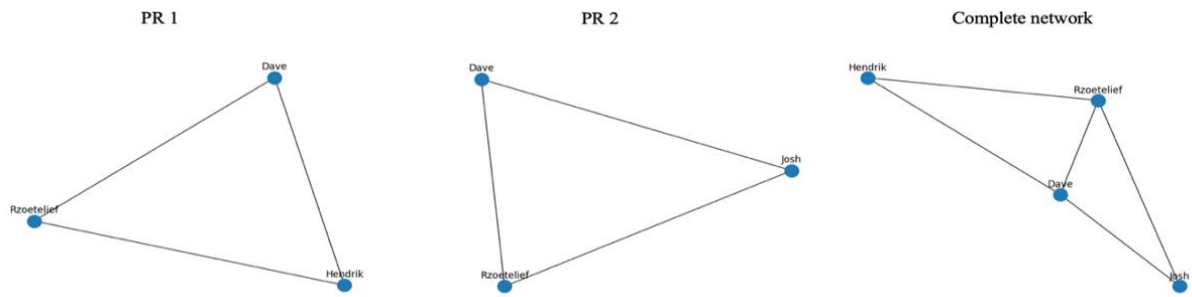


***Figure 3**: Construction of the social networks on GitHub*

Even though the pull request data shows in what direction nodes are connected with each other, this study constructed undirected graphs instead of directed graphs for several reasons (Oliveira, 2012). The construction process becomes easier because one treats all observations equally in an undirected graph and this would require less computational power compared to directed graphs. Additionally, undirected graphs used in the calculation of our metrics have no bearing on the final results. Furthermore, this makes the calculation of the measures simpler in terms of having a clearer insight into the structure and dynamics of the network.

The assumption is made that within the same transaction, each node is connected to the other, reflecting their engagement with each other around the pull request. This approach creates a detailed map of collaboration among nodes within repositories on a monthly basis, capturing the dynamics of interaction between users. By constructing the network in this matter, we can accurately represent the patterns of collaboration and engagement on GitHub, which is crucial for analyzing the collaboration patterns and social network.

Ultimately, in Python, these networks are created and grouped by repository ID, month and year. See figure 4 for a limited timeframe pre- and post-treatment for the repository called training-kit.

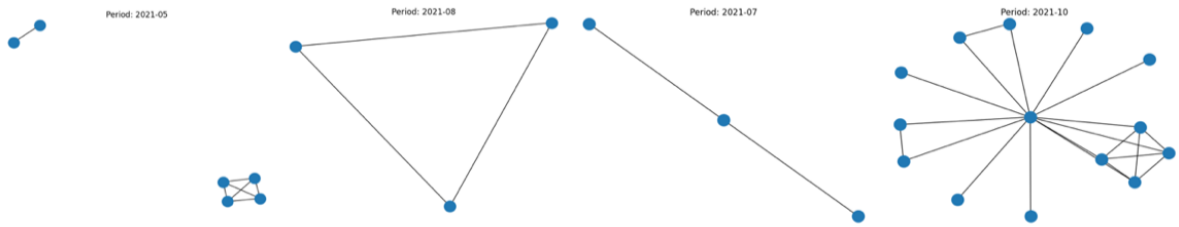


Figure 4: *Construction of repository training-kit*

3.3. Two cases

This study will treat the collected data as panel data to evaluate the integration of Artificial Intelligence (AI) into open source projects. To calculate and compare SNA metrics, it investigates the effect across two different levels (actor and network). Additionally, two distinct cases (GitHub and Google) are examined. The time span includes the technical preview phase, starting at the beginning of 2020 and continuing until the public release launch. Which allows for a comparison of the effects before and after the adoption of the AI tool for our treatment group GitHub. The timeframe is shown in figure 5.

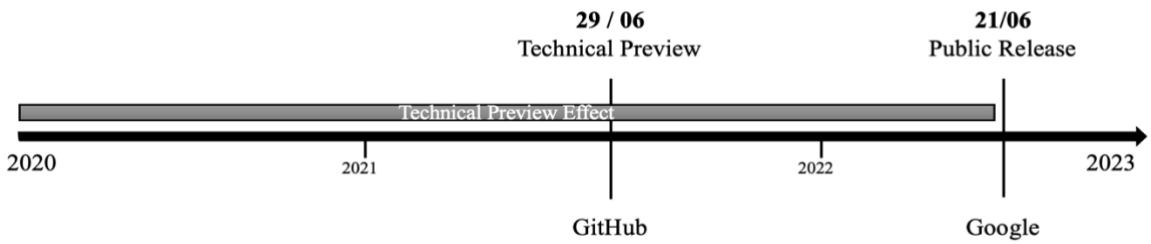


Figure 5: *Timeframe technical preview effect*

3.4. Difference-in-Differences Approach

Differences-in-differences (DiD) is a popular method for estimating causal effect in non-experimental setting (Roth, 2023). This approach will be used to determine the effect of the integration of AI on OSS projects. Within this study, the technical preview period is examined. GitHub serves as the treatment group, while Google, which was introduced to Copilot at a later date, acts as the control group. This setup allows us to compare changes in outcomes over time between these two groups. DiD approach enables the comparison of these changes, accounting for time-invariant unobserved characteristics as typically analyzed in panel data studies. This method effectively isolates the impact of specific interventions by comparing temporal changes across groups (Fredriksson, 2019).

3.5. Regression Models

To quantitatively evaluate the influence of AI integration on OSS projects within the context of SNA we adopt the widely used two-way fixed effects (TWFE) model for DiD analysis, as described by Roth et al. (2023). This model calculates the causal impact by calculating and comparing the average effects over time between two groups. Those exposed to an intervention (treatment) and those that are not (control). In this case, GitHub is acting as the treatment group, whereas Google functions as the control group.

Furthermore, this study conducts an actor- and network-level analysis of GitHub and Google. Here, centrality measures and network metrics are included. Additionally, pull request is incorporated as a control variable in the regression model, which allows controlling for the activity level and helps to distinguish between activity level and the influence of Copilot for different measures. See table 2 for the variables.

Table 2: *Variables actor-level and network-level*

Context	Variables
Treatment Actor-level	Average; Degree Centrality, Betweenness Centrality, Closeness Centrality
Control Actor-level	Average; Degree Centrality, Betweenness Centrality, Closeness Centrality
Treatment Network-level	Density, Average Clustering Coefficient, Average Path Length
Control Network-level	Density, Average Clustering Coefficient, Average Path Length
Control variable	Pull Request Count

The TWFE approach, however, assumes uniform treatment effects across different units and time periods; it does not account for potential changes in treatment effects over time. Given these constraints, this study does not explore the varied impacts of AI tools across different OSS communities. Instead, a focus is placed on specific instances where the AI introduction is consistent, simplifying the analysis by maintaining homogenous treatment timing. This is especially relevant for all the SNA variables, such as density, clustering coefficient, and average path length. Allowing for clear observation of the general effects of AI tools on these network characteristics without the complexity of varying treatment effects.

However, given the fact that TWFE assumes a homogenous treatment effect, it may not be as effective in capturing diverse impacts on the centrality measures unless data is segmented to account for within-group variations. As a result, to align with the two-way fixed effects, this study calculates the monthly average of centrality measures for each network. Averaging these measures helps to smooth out individual outliers and considers the fact that not every node will be present in every

network on monthly basis. Moreover, it would stay focussed on overarching trends within the network with the integration of AI, rather than specific individual nodes who might or might not be present in the subsequential network. This approach not only supports the TWFE model's assumption of homogeneous treatment effect, but it will also improve the statistical robustness of the analysis. By examining these monthly effects, this paper aims to get a clearer understanding of the developments in the OSS communities regarding social network dynamics and collaboration patterns. Therefore, the regression model that is employed, is specifically tailored to fit within this TWFE framework:

$$Y_{ist} = \alpha_i + \gamma_t + \beta_1 PR_{Count_{ist}} + \beta_2 Treated_{ist} + \beta_3 TechnicalPreview_{ist} + \beta_4 (Treated_{ist}\ x\ TechnicalPreview_{ist}) + \epsilon_{ist}$$

Where

- Y_{ist}: Dependent variable (e.g., Avg_Degree_Centrality).
- α_i: Individual Fixed Effects
- γ_t: Time-specific fixed effects.
- PR_Count_{ist}: The number of pull requests made by the actors.
- $Treated_{ist}$: A binary variable indicating if the observation is from GitHub (1) or Google (0).
- $TechnicalPreview_{ist}$: A binary variable indicating if the observation is from the period after the technical preview (1) or before (0).
- $Treated_{ist}\ x\ TechnicalPreview_{ist}$: The interaction term between treatment and technical preview.
- ϵ_{ist}: The error term, capturing random variation in $Yist$, not explained by the model.

The equation within the context of our study aims to find if the dependent variables social network metrics, will change due to the integration of Copilot for GitHub compared to Google. Where $\beta_1 PR_{Count_{ist}}$ represents the number of pull requests per observation. The presence of the treatment group is indicated by the binary variable $\beta_2 Treated_{ist}$, which takes the value of 1 for GitHub and 0 for Google. The variable $\beta_3 TechnicalPreview_{ist}$ indicates whether an observation is from the period after the intervention, in this case, the technical preview of Copilot, with 1 for the period after the preview and 0 for the period before. Additionally, the interaction term $\beta_4 Treated_{ist}\ x\ TechnicalPreview_{ist}$ combines the effects of being in the treated group and the time period after the technical preview.

The symbols i, s, and t represent individual actors, specific observations, and time periods. This means that the model accounts for variations across individuals, different data points for those individuals, and changes over time.

The TWFE model accounts for both individual fixed effects (α_i), which control for unobserved characteristics that are constant over time for each invidual, such as specific repository characteristics, and time fixed effects (γ_t), which represent temporal changes that effect all observations over time. These simplifications allow the model to crearly assess how AI integration on OSS projects influence social and collaboration dynamics over time, represented by months, using pre-calculated SNA metrics at either the actor or network level.

3.5.1. Hypotheses

3.5.1.1. Collaboration Network

Hypothesis 1*:* AI integration affects the collaboration network, potentially centralizing or decentralizing the network.

$$Average\ Degree\ Centrality_{it} = \alpha_i + \gamma_t + \beta_1 PR_{Count_{ist}} + \beta_2 Treated_{ist} + \beta_3 TechnicalPreview_{ist} + \beta_4 (Treated_{ist}\ x\ TechnicalPreview_{ist}) + \epsilon_{ist}$$

Hypothesis 2: AI integration will lead to a decrease in the average betweenness centrality of the network.

$$Average\ Betweenness\ Centrality_{it} = \alpha_i + \gamma_t + \beta_1 PR_{Count_{ist}} + \beta_2 Treated_{ist} + \beta_3 TechnicalPreview_{ist} + \beta_4 (Treated_{ist}\ x\ TechnicalPreview_{ist}) + \epsilon_{ist}$$

Hypothesis 3: AI integration will lead to an increase in the average closeness centrality of the network.

$$Average\ Closeness\ Centrality_{it} = \alpha_i + \gamma_t + \beta_1 PR_{Count_{ist}} + \beta_2 Treated_{ist} + \beta_3 TechnicalPreview_{ist} + \beta_4 (Treated_{ist}\ x\ TechnicalPreview_{ist}) + \epsilon_{ist}$$

3.5.1.2. Network structure

Hypothesis 4: AI integration in OSS will lead to an increase in density.

$$Density_{it} = \alpha_i + \gamma_t + \beta_1 PR_{Count_{ist}} + \beta_2 Treated_{ist} + \beta_3 TechnicalPreview_{ist} + \beta_4 (Treated_{ist}\ x\ TechnicalPreview_{ist}) + \epsilon_{ist}$$

Hypothesis 5: AI-integration in OSS will lead to an increase in the average clustering coefficient.

$$Average\ Clustering\ Coefficient_{it} = \alpha_i + \gamma_t + \beta_1 PR_{Count_{ist}} + \beta_2 Treated_{ist} + \beta_3 TechnicalPreview_{ist} + \beta_4 (Treated_{ist}\ x\ TechnicalPreview_{ist}) + \epsilon_{ist}$$

Hypothesis 6: AI-integration in OSS will lead to a decrease in the average path length.

$$Average\ Path\ Length_{it} = \alpha_i + \gamma_t + \beta_1 PR_{Count_{ist}} + \beta_2 Treated_{ist} + \beta_3 TechnicalPreview_{ist} + \beta_4 (Treated_{ist}\ x\ TechnicalPreview_{ist}) + \epsilon_{ist}$$

The variables are all defined in accordance with the statistical framework and hypotheses, Table 3 provides an overview.

Table 3: *Regression variables*

Variables	Description
Independent Variables	
Treated	Indicating presence of treatment group.
Technical Preview	Indicating presence of exposed to technical preview.
Treated x Technical Preview	Interaction term between treatment and technical preview.
Dependent Variables	
Degree Centrality	Measures node connectivity in network.
Betweenness Centrality	Tracks control over network's information flow.
Closeness Centrality	Indicates speed of accessing network nodes.
Density	Proportion of potential connections realized.
Clustering Coefficient	Extent of nodes' interconnectivity, forming cliques.
Average Path Length	Average distances between all node pairs.
Control Variable	
Pull Request Count	Number of pull requests.

For each hypothesis, the regression model is calculated using Stata, and the results are interpreted and compared to understand the effects of AI integration on OSS projects.

4. Results

This section presents the findings from the analysis. Firstly, a review is given for the cleaning and preparing of the data. Then, descriptive statistics are presented. Following this, the results of the parallel trends are discussed. Ultimately leading to the main results of the statistical analysis.

4.1. Data Cleaning and Preparation

4.1.1. Outliers Analysis

During the data analysis, several severe outliers across multiple variables were identified. Outliers are frequently encountered while collecting data, which can reduce the reliability of results. It also reduces data efficiency and introduces a large bias into the outcomes. These outliers were notably extreme and unreal, likely due to incorrect calculations in Python when computing the measures. Studies like Kwak et al. (2017) emphasize the necessity of addressing this issue. The identified outliers, a total of 502 observations, were replaced with the average of the mean values from both GitHub and Google networks. This approach allowed us to retain the total number of observations, which was crucial given the limited number of data points available. Omitting these observations entirely would have further reduced the sample size, potentially compromising the robustness and reliability of the analysis.

The primary reason behind these extreme outliers was linked to the construction of the network measures. Upon closer examination, A discovery was made that many constructed networks were not connected within a repository. This lack of connectivity likely led to the calculation of measures that were not applicable or realistic. Most of the social network analysis (SNA) metrics used in this study typically range from 0 to 1. However, the disconnected nature of these networks resulted in values that fell outside this expected range. The dataset's integrity could be preserved, and the reliability of the subsequent analysis was improved by substituting mean values for outliers.

4.1.2. Removal of Incomplete Observations

In addition to addressing outliers, a significant number of observations were removed due to the lack of sufficient data. 2,767 observations, where more than four variables were empty or indicated as zero, were excluded from the statistical analysis. This step was necessary to ensure the quality and reliability of the data. Including such incomplete observations could lead to biased or inaccurate results, as the absence of zero values might distort the analysis (Kwak, 2017).

4.2. Descriptive Statistics

The descriptive statistics for both groups are presented in Table 4 and Table 5, respectively, covering observations from the start of 2020 until the end of July 2022. It is important to note the significant difference in the number of observations between GitHub and Google. This difference

arises from the larger volume of pull request data available for Google compared to GitHub. Apart from this, the descriptive measures of the variables appear quite similar between the two groups.

Table 4: *Descriptive statistics GitHub*

GitHub					
Variable	**Observations**	**Mean**	**Std. Dev.**	**Min**	**Max**
Avg_Degree_Centrality	838	0.717698	0.313089	0.0095656	1
Avg_Betweenness_Centrality	838	0.0605625	0.0986475	0	0.333333
Avg_Closeness_Centrality	838	0.8099672	0.235539	0.1555556	1
Density	838	0.7197618	0.313614	0.0095656	1.31338
Avg_Clustering_Coefficient	838	0.4486482	0.4233635	0	2.3067
Avg_Path_Length	731	1.242078	0.3362158	1	2.842208
PR_Count	838	39.53461	115.1027	2	1070

Table 5: *Descriptive statistics Google*

Google					
Variable	**Observations**	**Mean**	**Std. Dev.**	**Min**	**Max**
Avg_Degree_Centrality	3,989	0.7590317	0.2766065	0.002097	1
Avg_Betweenness_Centrality	3,989	0.0627614	0.095923	0	0.3333333
Avg_Closeness_Centrality	3,989	0.8492892	0.1961159	0.010989	1
Density	3,989	0.7597258	0.2773711	0.002097	1.31338
Avg_Clustering_Coefficient	3,989	0.4540545	0.4306082	0	2.3067
Avg_Path_Length	3,778	1.22867	0.2875217	1	2.841667
PR_Count	3,989	31.11682	111.8373	1	2000

4.3. Parallel Trends Test

For the DiD approach to be valid, it is essential that the parallel trends assumption holds, meaning that the treatment and control groups must exhibit similar trends in the pretreatment period (Ryan, 2015). This is crucial because the control group serves as the "counterfactual," representing the unobserved outcome of the treatment group had no intervention occurred. Since the parallel trends assumption involves an unobservable counterfactual, it cannot be proven; however, reasonable evidence can support it. To test the parallel assumption, the similarity of linear trends between the treatment group and control groups are examined through statistical analysis utilising a treated time interaction term (Ryan, 2015). Here, the binary variable 'treated' is indicated with either the treatment

group (1) or the control group (0) and 'time' refers to the period before the introduction of the technical preview. Table 6 presents the results of this parallel test for the pretreatment period.

Table 6: *Parallel trend test results*

Variable	Interaction Term	Coefficient	Std Err.	P-value
Avg_Degree_Centrality	treated: time	-.0006363	.0028056	0.821
Avg_Betweenness_Centrality	treated: time	.0003258	.0008395	0.698
Avg_Closeness_Centrality	treated: time	-.0002268	.0019832	0.909
Density	treated: time	-.0013196	.0026778	0.622
Avg_Clustering	treated: time	.0113265	.003524	0.001***
Avg_Path_Length	treated: time	.002662	.0032596	0.414
PR_Count	treated: time	.5550582	1.094782	0.612

Note: *p<0.1; **p<0.05; ***p<0.01

The results indicate that for most variables, there is no significant difference in trends between the treatment and control groups in the pretreatment period, which is identified by the treated variable not having statistically significant p-values for our social network metrics and pull request count metrics. However, it is important to note that the p-value for the average clustering coefficient is p<0.001. This means there is strong evidence against the null hypothesis (which states that the coefficient is zero), suggesting that the interaction term has a statistically significant effect on the dependent variable Avg_Clustering. This indicates that the effect of time on the average clustering coefficient is different for the treatment group compared to the control group. Therefore, it is essential to approach this variable with caution when interpreting the outcomes.

4.4. Main Analysis Results

Within this section a description is given for all regression models used in the analysis with the adoption of the two-way fixed effects (TWFE) model for the DiD analysis. See table 7 for a summary of the DiD model, examining its relationship with the interaction between treatment and technical preview.

Table 7: *Main results*

Dependent Variable	Coefficient	Std Err.	P-value	R-squared
Avg_Degree_Centrality	-.006615	.0200327	0.741	0.5595
Avg_Betweenness_Centrality	.0011479	.0079902	0.886	0.2460
Avg_Closeness_Centrality	-.0020516	.0151706	0.892	0.5130
Density	-.0116017	.0209683	0.580	0.5511

Avg_Clustering	.0564138	.0312898	0.072	0.3694
Avg_Path_Length	.0312104	.0251279	0.215	0.5816

It is notable that almost none of the dependent variables show a statistically significant relationship with the interaction term between treatment and technical preview. The p-values for most of the variables are above conventional significance levels ($p > 0.05$ or 0.10), indicating that the interaction term does not significantly affect these social network metrics. The only variable that shows results below the $p<0.10$ threshold is Avg_Clustering. But, overall, this indicates that the introduction of AI Copilot on GitHub does not significantly alter these social network dynamics compared to the control group.

It is essential to approach the interpretation of these results with caution, especially considering the p-value for Avg_Clustering below to the $p<0.10$ threshold, indicating a potential area for further research. However, as mentioned in the section above, this variable could also have been influenced by fixed effects. The results suggest that the AI integration of Copilot does not have a statistically significant effect on most of the social network metrics. Table 8 presents our support for the hypotheses.

Table 8: *Support for hypotheses*

Hypothesis	**Dependent Variable**	**Support for Hypothesis**
H1a: AI integration in OSS will lead to greater centralization of the OSS network.	Avg_Degree_Centrality	No
H1b: AI integration in OSS will lead to greater decentralization of the OSS network.	Avg_Degree_Centrality	No
H2: AI integration in OSS will lead to a decrease in the average betweenness centrality of the network.	Avg_Betweenness_Centrality	No
H3: AI integration in OSS will lead to an increase in the average closeness centrality of the network.	Avg_Closeness_Centrality	No
H4: AI integration in OSS will lead to an increase in density.	Density	No
H5: AI integration in OSS will lead to a increase in the average clustering coefficient.	Avg_Clustering	Only at 0.10 threshold
H6: AI integration in OSS will lead to a decrease in the average path length.	Avg_Path_Length	No

5. Discussion

Artificial Intelligence (AI) has become a prominent topic in open source communities. This technological advancement is reshaping the landscape of software development (Zohair, 2018), shifting the focus from traditional coding practices to a model where new contributors, in collaboration with AI, primarily understand and utilize code for software creation (Bird, 2022). Such a paradigm shift raises questions about how social network dynamics and collaboration patterns may evolve. Therefore, this research specifically investigates how the increasing adoption of AI Copilot (Kharuffa, 2023) will introduce different approaches in software creation (Savary-Leblanc, 2022) and what effect it could have on various social aspects for GitHub communities.

5.1. Findings

The study adopted the two-way fixed effects (TWFE) regresssion model for DiD analysis (Roth, 2023). The key goal for this research is to find out if the integration of AI in open source software (OSS) projects impact the social network dynamics and collaboration patterns, compared to traditional OSS projects. The empirical results reveal that none of the dependent variables show a statistically significant relationship. This indicates that there is no support for the theorization that the introduction of AI Copilot significantly alters social network dynamics compared to the control group. Therefore, suggesting that the anticipated impacts of AI integration on social network dynamics and collaboration patterns may not be as pronounced or may require further data and refined analysis to detect.

The statistical analysis yielded several unexpected results. Most of the key metrics of the social network analysis contained a notably high p-value, above a $p<0.05$ significance threshold. For instance, average degree centrality has a coefficient of (-0.006615) with a p-value of (0.741), suggesting no significant change post-intervention. Similarly, betweenness centrality (coefficient 0.0011479, p-value 0.886) and density (coefficient -.0116017, p-value 0.580) also show no significant variations. The only metric approaching significance is average clustering (coefficient 0.0564138, p-value 0.072), indicating a potential and at a $p<0.10$ threshold significantly increase. However, this variable should be approached with caution, as the paralell trend may be violated for this specific variable. The results are unexpected, particularly given the intervention of Copilot, which was hypothesized to impact these network metrics significantly. Previous research in structural intervention in networks reported more pronounced changes. For instance, studies by Sun (2023) demonstrated significant impacts of structural interventions on network efficiency and collaborations patterns.

5.2. Implications

The insignificant results of social network analysis provide nonetheless several relevant contributions to the academic field. The theoretical framework presents an initial review of existing theories which analyses how AI can potentially affect the social network dynamics and collaboration patterns on OSS repositories on GitHub. Combined with the conceptual framework, it constitues an interesting and novel view for others scholars to built further upon. Moreover, by examining pull request data, this study exposes potential pitfalls on the reliance on a single type of interaction with definining the edges for these complex networks. Li et al. (2022) ackownledges this insight as well that while pull request offer rich and valuable data, relying solely on PR data present challenges, where it can miss important interactions to capture the real social network dynamics. In addition, the usage of averaging techniques for certain metrics has revealed the potential that such methods can potentially hide significant variations in network metrics. This reflection is valuable towards future research in social network analysis, where this study suggest more nuanced aggregation methods might be necessary to capture the true dynamics of community interactions. Studies like those of (Borgatti, 2024; Lee J. B., 2021) support the same view that temporal granularity can significantly impact the interpretation of social network data. Finally, this is one of the first attempts to capture the effects of the introduction of Copilot through the lens of social network analysis on GitHub's social network dynamics and collaboration patterns. This approach provides a valuable foundation for future research. By exploring existing theories, forming methodologies and identifying potential challenges, this study provides insights for subsequent research to refine these techniques and delve further into this area. Researchers can build on this foundation to develop more comprehensive models and analyses to better capture the dynamics of software development communities.

5.3. Limitations

A possible explanation for these insignificant results is the limited amount of data being used. The analysis was restricted to a smaller subset of pull request data from GitHub, possibly insufficient to capture impacts of the introduction of Copilot. Larger datasets typically provide more robust insights, reducing noise and variability than alter effects. This limitation align with findings by Dhawan et al. (2021), who highlights the need for large datasets in social network analysis for accurately defining and evaluating network communities. Moreover, the method of averaging node activity could also weaken significant variations. As discussed in the methodology section, averaging can smooth out the variables which in return hides true variability and dynamics over the constructed networks. In other words, making it challenging to detect true changes. Freeman (2004) highlights that centrality measures are sensitive to network change. The involvement of compiling data from multiple points into a single aggregated measure, or within this study, taking the average of certain metrics is also discussed by Borgatti et al. (2024). The authors mention that it could potentially vague

significant network dynamics by smoothing out these short-term variations. Lastly, the introduction of Copilot may have had a more subtle or complex effect than anticipated. Network interventions often lead to multifaceted changes that are not immediately seen in simple metrics. The study by Lima et al. (2014) suggest that the impact of interventions and user interactions develop progressively, requiring extended observation to fully understand their significance. Meaning that, within the context of this study, the introduction of Copilot perhaps could take a much longer time to manifest significantly.

5.4. Future Research

Further research is needed to determine whether the introduction of Copilot could have a significant impact on network dynamics and collaboration patterns on the GitHub platform. Based on the conducted research, future studies within this topic should take several factors into account. To begin with, it is recommended to gather a larger amount of diverse data in order to build social networks. This means not relying solely on pull request data, as the nature of this data can make network constructing challenging. This difficulty arises due to the fact that pull requests are not present every month in the repositories, leading to either the absence of network or the construction of non-interconnected networks. Additionally, through these unconnected networks a significant amount of extreme outliers was identified per observation. Which, given the nature of the aforementioned networks, is most likely the result of errors made during the network's construction and analysis in Python. Therefore, it is recommended to establish a more comprehensive network which goes beyond solely pull request data to incorporating various types of relational data like issues and commits. Besides incorporating an increasing amount of various data, future studies should consider the issue of averaging multiple metrics for the calculations of the SNA metrics (i.e., average centrality measures and average clustering coefficient). As mentioned in the limitations section, temporal aggregation might influence our metrics. The research of Borgatti et al. (2024) revealed that some approaches could more accurately capture the true measures by using finer temporal granularity. These methods could use overlapping time windows to maintain some temporal granurality while smoothing out the data, or they could be applied on a weekly basis to capture short-term variations and more immediate impact of interventions. However, it is essential to keep in mind that these results could also be the consequence of lack of data in creating the networks.

6. Conclusion

This study explored the potential impact of Artificial Intelligence (AI) Copilot before and after the technical preview of June 2021 on various repositories from GitHub. The study compared these effects with those observed from Google repositories, which were exposed to AI Copilot at a later stage. These effects were measured through multiple social network metrics to capture the social network dynamics and collaboration patterns within these communities. However, the main research question remains unanswered due to several reasons presented in detail in the results and discussion section.

Nonetheless, during the process, several important insights emerged. Firstly, significant attention must be paid to the data collection. Collecting pull request data initially made the most sense because of their more collaborative nature during the development process on GitHub. But they are less frequent than issues or commits, which resulted in a lot of networks that were disconnected or did not exist in that particular month at all. Therefore, additional data sources like issues and commits could provide a more comprehensive view of the network. Subsequent research should consider collecting more pull request data or adding other sources. However, due to time constraints, it was not possible to gather more for this study. In addition to data collection and network construction, the social network metrics could potentially be calculated in different ways. Perhaps instead of taking the average at node level, keep track of individual nodes and see how they develop over time before and after the intervention of Copilot. Alternatively, creating different temporal granular effects can possibly have an impact on the results.

Open source software project managers, developers, and contributors incorporating AI into their projects can use this study as a foundation for understanding the creation and construction of networks within AI context. Future studies can build upon, confirm, or enrich these findings, addressing the identified limitations and expanding knowledge in this dynamic and volatile environment of open source communities.

Understanding the potential effects AI may have on GitHub and other OSS platforms is critical as it continues to impact the industry. GitHub's social structure encourages cooperation among communities in software development. The introduction of Copilot, which acts as an assistant or even replacement for coding tasks, accelerates the dynamics in ways that developers must closely monitor. There is still a lot to explore within this environment, particularly concerning the social aspects. In conclusion, this research opens new avenues for future studies aimed at enhancing our understanding of the social network dynamics and collaboration patterns between AI and developers. Such insights will be vital in navigating new networks within the AI revolution in OSS.

Bibliography

A. Mashkoor, T. M. (2022). Artificial intelligence and software engineering: Are we ready? *Computer*, 24 - 28.

Bird, C. F. (2022). Taking Flight with Copilot: Early insights and opportunities of AI-powered pair-programming tools. *Queue*, 35 - 57.

Blumberg, B. C. (2014). *EBOOK: Business research methods.* McGraw Hill.

Bonaccorsi, A. &. (2003). Why Open Source software can succeed. *Research Policy*, 1243 - 1258.

Borgatti, S. P. (2024). *Analyzing social networks.* SAGE Publications.

Bosch, J. (2009). From software product lines to software ecosystems. *From software product lines to software ecosystems*.

Božić, V. &. (2023). Chat GPT and education. *Preprint*.

Brown, T. M. (2020). Language models are few-shot learners. *Advances in neural information processing systems*, 1877 - 1901.

Chen, M. T. (2021). Evaluating Large Language Models Trained on Code . *Cornell University*.

Chittibala, D. R. (2024). Advancements in Automated Code Scanning Techniques for Detecting Security Vulnerabilities in Open Source Software. *International Journal of Computing and Engineering*, 16 - 25.

Concas, G. L. (2008). Open Source Communities as Social Networks: an analysis of some peculiar characteristics. *19th Australian Conference on Software Engineering* (pp. 387 - 391). IEEE.

Crawford, T. D. (2023). AI in Software Engineering: A Survey on Project Management Applications.

Crowston, k. S. (2002). Open source software projects as virtual organisations: competency rallying for software development. *IEEE Proceedings - Software*, 3 - 17.

Dabbish, L. S. (2012). Social coding in GitHub: transparency and collaboration in an open software repository. *Proceedings of the ACM 2012 conference on computer supported cooperative work*, (pp. 1277 - 1386).

Dakhel, A. M. (2023). GitHub Copilot AI pair programmer: Asset or Liability? *Journal Of Systems And Software/□The □Journal Of Systems And Software*.

De Chaisemartin, C. &. (2023). Two-way fixed effects and differences-in-differences with heterogeneous treatment effects: A survey. . *The Econometrics Journal*, C1 - C30.

Degenne, A. &. (1999). *Introducing Social Networks.* London: Sage Publications.

Dhawan, S. K. (2021). Defining and Evaluating Network Communities Based on Ground-Truth in Online Social Networks. *Recent Innovations in Computing: Proceedings of ICRIC 2020.* Singapore: Springer.

Dias, L. S. (2018). Who drives company-owned OSS projects: internal or external members? *Journal of the Brazilian Computer Society*.

Dohmke, T. (2022, June 22). *The GitHub Blog*. Opgehaald van GitHub: https://github.blog/2022-06-21-github-copilot-is-generally-available-to-all-developers/

Dohmke, T. (2023, November 8). *GitHub*. Opgehaald van GitHub blog: https://github.blog/2023-11-08-universe-2023-copilot-transforms-github-into-the-ai-powered-developer-platform/

Duijn, V. &. (2006). What is special about social network analysis. *Methodology*.

Evers, S. (2000). An Introduction to Open Source Software Development. *Technische Universität Berlin, Fachbereich Informatik, Fachgebiet Formale Modelle, Logik und Programmierung (FLP)*.

Fredriksson, A. a. (2019). Impact evaluation using Difference-in-Differences. *RAUSP Management Journal*, 519 - 532.

Freeman, L. C. (2004). *The development of social network analysis: A study in the sociology of science.* Vancouver : Emperical Press.

Friedman, N. (2021, June 29). *Introducing GitHub Copilot: your AI pair programmer*. Opgehaald van The GitHub Blog: https://github.blog/2021-06-29-introducing-github-copilot-ai-pair-programmer/

Gerber, A. M. (2010). Documenting open sourcing migration processes for re-use. *Proceedings of the 2010 Annual Research Conference of the South African Institute of Computer Scientists and Information Technologists*, 75 - 95.

GitHub. (2001, May 2). *GitHub GraphQL API documentation*. Opgehaald van GitHub Docs: https://www.scribbr.com/citation/generator/folders/4H6dXOHFOiME7SGMnvxu8F/lists/7jluzJNIQgvPGMWoDBzfTn/

GitHub. (2023, January 25). *100 million developers and counting*. Opgehaald van GitHub: https://github.blog/2023-01-25-100-million-developers-and-counting/

GitHub. (sd). *About pull requests*. Opgehaald van GitHub Docs: https://docs.github.com/en/pull-requests/collaborating-with-pull-requests/proposing-changes-to-your-work-with-pull-requests/about-pull-requests

Gottlander, J. &. (2023). The Effects of AI Assisted Programming in Software Engineering.

Granovetter, M. (1973). The strenght of Weak Ties. *American Journal Of Sociology*, 1360 - 1380 .

Gunnell, L. N. (2024). Equation-based and data-driven modeling: Open-source software current state and future directions. *Computers & Chemical Engineering*, 181.

Hazmi Hassri, M. &. (2023). The Impact of Open-Source Software on Artificial Intelligence. *Journal of Mathematical Sciences and Informatics*.

He, P. L. (2012). Applying centrality measures to the behavior analysis of developers in open source software community. *2012 Second International Conference on Cloud and Green Computing*, (pp. 418 - 423). Xiangtan.

Horta, V. A. (2022). Detecting topic-based communities in social networks: A study in a real software development network. *Journal of Web Semantics*.

Hossain, L. &. (2009). Social networks and coordination performance of distributed software development teams. *The Journal of High Technology Management Research*, 52 - 61.

Iacus, S. M. (2012). Causal inference without balance checking: Coarsened Causal inference without balance checking: Coarsened exact matching. matching. *Political analysis*, 1 - 24.

Imai, S. (2022). Is github copilot a substitute for human pair-programming? an empirical study. *InProceedings of the ACM/IEEE 44th International Conference on Software Engineering: Companion Proceedings*, (pp. 319 - 321).

Jansen, S. C. (2013). *Software ecosystems: Analyzing and managing business networks in the software industry.* Edward Elgar Publishing.

Jansen, S. F. (2009). Sense of community: A research agenda for software ecosystems. *Proceedings of the 31st Inter- national Conference on Software Engineering* (pp. 187 - 190). Companion Volumne.

Jeffrey A. Roberts, I.-H. H. (2006). Understanding the Motivations, Participation, and Performance of Open Source Software Developers: A Longitudinal Study of the Apache Projects. *Management Science*.

Kabakus, A. T. (2020). Githubnet: Understanding the characteristics of github network. *Journal o Web Engineering*, 557 574.

Kalliamvakou, E. G. (2014). The promises and perils of mining GitHub. *In Proceedings of the 11th working conference on mining software repositories* (pp. 92 - 101). ACM.

Kharrufa, C. B. (2023). Generative Artificial Intelligence Assistants in Software Development Education: A Vision for Integrating Generative Artificial Intelligence Into Educational Practice, Not Instinctively Defending Against It. *IEEE Software*, 52 - 59.

Kilamo, T. H. (2012). From proprietary to open source—Growing an open source ecosystem. *The Journal of Systems and Software*, 1467 - 1478.

Kilamo, T. H. (2012). From proprietary to open source—Growing an open source ecosystem. *Journal of Systems and Software*, 1467 - 1478.

Kochhar, P. S. (2021). Moving from Closed to Open Source: Observations from Six Transitioned Projects to GitHub. *IEEE Transactions on Software Engineering*, 1838 - 1856.

Kwak, S. K. (2017). Statistical data preparation: management of missing values and outliers. *Korean journal of anesthesiology,*, 70.

Latorre, R. S. (2017). Measuring social networks when forming information system project teams. *The Journal of Systems and Software*, 304 - 323.

Lee, G. K. (2003). From a firm-based to a community-based model of knowledge creation: the case of the Linux kernel development. *Organization Science*, 633 - 649.

Lee, J. B. (2021). Dynamic node embeddings from edge streams. *IEEE Transactions on Emerging Topics in Computational Intelligence* (pp. 931 - 946). IEEE.

Li, N. Y. (2022). Artificial intelligence capability and organizational creativity: the role of knowledge sharing and organizational cohesion. *Frontiers in Psychology*, 13.

Li, Z. Y. (2022). Opportunities and challenges in repeated revisions to pull-requests: An empirical study. *Proceedings of the ACM on Human-Computer Interaction* (pp. 1 - 35). New Yorkj: Association for Computing Machinery.

Lima, A. R. (2014). Coding together at scale: GitHub as a collaborative social network. *In Proceedings of the international AAAI conference on web and social medi*, (pp. 295 - 304).

Lumbard, K. G. (2024). An empirical investigation of social comparison and open source community health. *Information Systems Journal*, 499 - 532.

Madey, G. F. (2002). The open source software development phenomenon: An analysis based on social network theory. *Americas Conference on Information Systems (AMCIS)* . Association for Information Systems (AIS).

Manikas, K. &. (2013). Software ecosystems–A systematic literature review. *Journal of systems and Software*, 1294–1306.

Martínez-Torres, M. R. (2015). A quantitative study of the evolution of open source software communities. *World Academy of Science, Engineering and Technology*, 1374 - 1379.

McClean, K. G.-L. (2021). Social network analysis of open source software: A review and categorisation. *Information and Software Technology*.

Messerschmitt, D. G. (2003). Software ecosystem: understanding an indispensable technology and industry. *MIT Press*.

Moradi-Jamei, B. K. (2021). Community formation and detection on GitHub collaboration networks. *Proceedings of the 2021 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining*, (pp. 244 - 251).

Newman, M. (2010). *Networks: An Introduction.* Oxford: Oxford Academic.

Oliveira, M. &. (2012). An overview of social network analysis. *WIREs Data Mining Knowl Discov*, 99 - 115.

Otte, E. &. (2002). Social network analysis: a powerful strategy, also for the information sciences. *Journal of Information Science*, 441 - 453.

Pammer-Schindler, M. L. (2021). Automation and Artificial Intelligence in Software Engineering: Experiences, Challenges, and Opportunities. *Proceedings of the 54th Hawaii International Conference on System Sciences*, (pp. 146 - 156).

Puryear, B. a. (2022). Github copilot in the classroom: learning to code with AI assistance. *Journal of Computing Sciences in Colleges*, 37 - 47.

Rathee, S. &. (2022). *Getting Started with Open Source Technologies: Applying Open Source Technologies with Projects and Real Use Cases.* Apress eBooks.

Red Hat. (2022). *The State of Enterprise Open Source.* Red Hat.

Roth, J. S. (2023). What's trending in difference-in-differences? A synthesis of the recent econometrics literature. *Journal of Econometrics*, 2218 - 2244.

Ryan, A. B. (2015). Why We Should Not Be Indifferent to Specification Choices for Difference-in-Difference. *Health Services Research*, 1211–1235.

Savary-Leblanc, M. B. (2022). Software assistants in software engineering: A systematic mapping study. *Software, Practice & Experience/Software, Practice And Experience*, 856 - 892.

Schreiber, R. (2023). Organizational Influencers in Open-Source Software Projects. International Journal of Open Source Software and Processes. *International Journal of Open Source Software Processes*.

Shah, V. (2019). Towards Efficient Software Engineering in the Era of AI and ML: Best Practices and Challenges. *International Journal of Computer Science and Technology*, 63 - 78.

Sheng, J. D. (2020). Identifying influential nodes in complex networks based on global and local structure. *Physica A: Statistical Mechanics and its Applications*.

Sheng, J. D. (2020). Identifying influential nodes in complex networks based on global and local structure. *Physica A: Statistical Mechanics and its Applications*.

Sigfridsson, A. &. (2011). On qualitative methodologies and dispersed communities: Reflections on the process of investigating an open source community. *Information and Software Technology*, 981 - 993.

Sobania, D. B. (2022). Choose your programming copilot: a comparison of the program synthesis performance of github copilot and genetic programming. *Proceedings of the genetic and evolutionary computation conference* (pp. 1019 - 1027). GECCO.

Steinmacher, I. P. (2018). Almost there: A study on quasi-contributors in open source software projects. *In Proceedings of the 40th international conference on software engineering* , (pp. 256 - 266).

Sun, Y. Z. (2023). STRUCTURAL INTERVENTIONS IN NETWORKS. *International Economic Review*, 1553 - 1563.

Sundaresan, S. &. (2022). AI-enabled knowledge sharing and learning: redesigning roles and processes. *International Journal of Organizational Analysis*.

Tan, X. Z. (2020). A first look at good first issues on GitHub. *In Proceedings of the 28th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering* (pp. 398 – 409). Association for Computing Machinery (ACM).

Teixeira, J. G.-B. (2015). Lessons learned from applying social network analysis on an industrial Free/Libre/Open Source Software ecosystem. *Journal of Internet Services and Applications* .

Torres, M. M. (2011). Analysis of the core team role in open source communities. *In 2011 International Conference on Complex, Intelligent, and Software Intensive Systems*, (pp. 109 - 114).

Van Antwerp, M. &. (2010). The importance of social network structure in the open source software developer community. *Proceedings of the 43rd Hawaii International Conference on System Sciences.*

Von Krogh, G. (2003, April 2003). *MIT Sloan Management Review*. Opgehaald van MIT Sloan Management Review: https://sloanreview.mit.edu/article/opensource-software-development/

Wang, D. W. (2019). Human-AI collaboration in data science: Exploring data scientists' perceptions of automated AI. *Proceedings of the ACM on human-computer interaction* (pp. 1 - 24). CSCW.

Wang, J. (2012). Survival factors for Free Open Source Software projects: A multi-stage perspective. *European Management Journal*, 352 - 371.

Washizaki, H. (2020). Towards software value co-creation with AI. *2020 IEEE 44th annual computers, software, and applications conference* (pp. 1117 - 1118). Madrid: COMPSAC.

Wasserman, S. F. (1994). *Social network analysis: Methods and applications.* New York: Cambridge University Press.

Wu, J. H. (2023). Social-technical network effects in open source software communities: Understanding the impacts of dependency networks on project success. *Information Technology & People*.

Zanetti, M. S. (2012). A Quantitative Study of Social Organization in Open Source Software Communities.

Zohair, L. M. (2018). The Future of Software Engineering by 2050s: Will AI Replace Software Engineers? *International Journal of Information Technology*, 1 - 13.

Zöller, N. J. (2020). A topology of groups: What GitHub can tell us about online collaboration. *Technological Forecasting and Social Change*.

Appendices

A. Utilization of AI Tools

During the development of this research, two AI tools have played an important factor. Therefore, to remain as transparent as possible this section entails what tools are utilized and how they are utilized during the process of conducting this study.

Chat GPT – Writing: First of all, Chat GPT is used as assistance during the writing process of this thesis. Chat GPT is a type of Generative Pre-trained Transformer (GPT), this large language model (LLM) is able respond to natural language inputs to generate text (Božić, 2023).

The LLM is a powerful tool, it helped me improving writing in an academic manner. However, I knew from the beginning when I was conducting this research that it would be beneficial for my research. This helped me in rewriting material to be able to present my own research more effectively, but every element of this study is based on my own research and input. In addition, I sought assistance from Chat GPT in situations when I was unsure whether or a grammar, spelling or stylistic errors was present or not.

GitHub's Copilot – Programming: Ironically speaking, I also used another GitHub's Copilot tool for the programming part. It was especially important because, at the beginning of the semester, I was only a very basic Python programmer. I gained a lot of experience by using the GitHub platform and the installing the extension Copilot within my Visual Studio Code environment. Which helped me in developing myself on a more advanced level. Puryear and Sprint (2022) discuss AI-driven development environments (AIDE) like Copilot, and how its features likely are expected to become the new completion standard. However, it also encourages computer science programmers to become familiar with how it works while avoiding complete reliance on these resources.

In the process, I realized that I required Copilot's assistance, especially because I did not have enough time to complete this portion with my own programming skills. This included constructing the networks, calculating the social network metrics and merging everything together. Despite this, I did my best to able to understand what I was doing and checking everything that I received as output from the AI pair programmer.

Additional contributions: Beyond these main applications, Chat GPT helped me to overcome some conceptual obstacles by presenting creative and innovative ideas. It was also very helpful in analysing and clarifying STATA outputs, which greatly helped in the completion to my thesis process.

Challenges: While GPT and Copilot have possible benefits with it, scholars should also be aware of its potential limitations. Where I would like to express that GPT could lead to bias and inaccuracies. Moreover, this technology dependence of AI tools could negatively influence critical thinking skills and creates independence in learning. Being aware of such negative impacts I have

tried my best to form my own opinion on my master's thesis and do not use these tools as a primary solution for different challenges that were encountered during the process of writing and research this study.

Whereas Copilot can save a great deal of time by suggestion snippets, functions, or even whole files. Copilot may also result in poor coding practises that produce inefficient or insecure recommendations. In conclusion, the AI coding assistant could be useful for writing code, but just as Chat GPT, scholar should be mindful of the suggestions and how important it is to balance the benefits and its drawbacks.

B. Computations of Measures

See below here for the computations of the measures for the Social Network Metrics.

Collaboration network

Average Degree Centrality: This is equal to the number of edges a node has with other nodes. The equation can be expressed as follows:

$$Average\ Degree\ Centrality = \frac{1}{n}\sum_{i=1} n\ \frac{\deg\ (i)}{n-1}$$

Where

- $Deg(i)$ is the number of direct connection (edges) for node i.
- n is the total number of nodes in the network.

Average Betweenness Centrality: Measures how important a node is in a network. Based on how many of the shortest path between any two nodes in the network pass through the node in question:

$$Average\ Betweenness\ Centrality = \frac{1}{n}\sum_{i=1} n\ (\frac{\sum j<k \frac{gjk^{(i)}}{g_{jk}}}{\frac{1}{2}n(n-1)})$$

Where

- n is the total number of nodes in the network.
- $G_{jk}(i)$ is the number of shortest paths between node j and node k that pass through node i.
- G_{jk} is the number of shortest paths between node j and node k.
- The inner sum is taken over all pairs of nodes j and k where $j < k$, ensuring each pair is only counted once for an undirected graph.
- The outer sum averages this value over all nodes i in the network.

Closeness centrality: Measures the distance of a node to all other in the network by focusing on the geodesic distance from each node to all others. It will ultimately show how long it will take information to spread from a given node to others in the network. See formula below:

$$Average\ Closeness\ Centrality = \frac{1}{n}\sum_{i=1} n\ \frac{n-1}{\sum_{u \neq i} d(i,u)}$$

Where

- $n-1$ represents the total number of other nodes (excluding i) in the network, ensuring normalization of the measure.
- $d(i,u)$ is the distance between i and u.
- The denominator for each i is the sum of the distances from i to all other nodes u in the network.
- n is the total number of nodes in the network.

A higher closeness centrality means that a node is generally closer to all other nodes. If a developer in a project has a high score, it implies they can quickly and efficiently communicate with other developers.

Network structure

Network Density: Describes the proportion of potential connections in a network that are actual connections. This formula for density can be applied in one of two ways, the 'ego-centric' approach or 'socio-centric' approach (Hossain, 2009). We will the use socio-centric approach however, because we want to understand the overall connectivity within a network, therefore:

$$Network\ Density = \frac{2\ x\ Number\ of\ Actual\ Edges}{n(n-1)}$$

Where

- n is the total number of nodes in the network.
- The numerator $2\ x\ Number\ of\ Actual\ Edges$ represents the actual number of edges in the network. The multiplication by 2 is necessary because each edge is counted twice in an undirected graph – one for each direction.
- The term $n(n-1)$ in the denominator represents total number of possible edges in an undirected network.

Average Clustering Coefficient: The clustering coefficient of a node measures how close its neighbors are to being a complete graph (clique):

$$Average\ Clustering\ Coefficient = \frac{1}{n}\sum_{i=1} n\ C_i$$

Where C_i is the local clustering coefficient for node i, which can be calculated as:

$$C_i = \frac{2T_i}{k_i(k_i - 1)}$$

Where

- T_i is the number of triangles through node i and k_i is the degree of node i.

Average Path Length: The average number of steps along the shortest paths for all possible pair of network nodes. It is a measure of information efficiency.

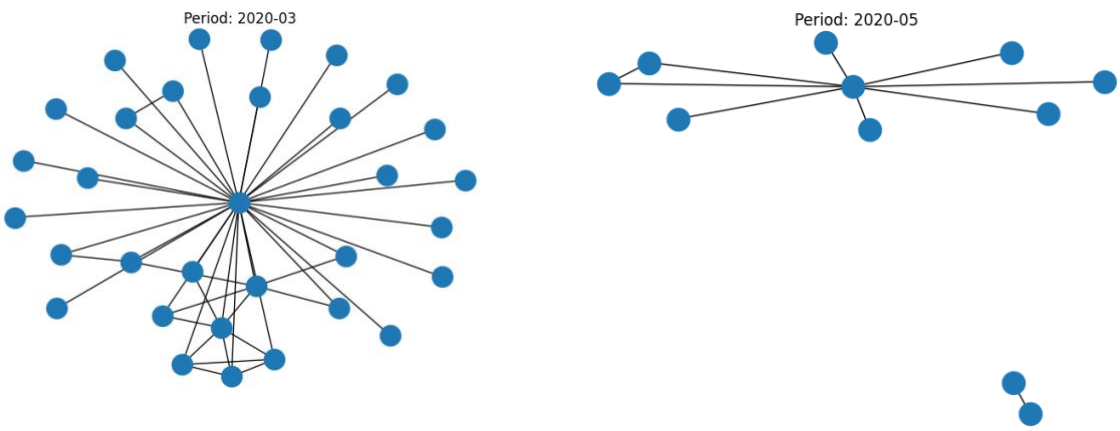
$$Average\ Path\ Length = \frac{1}{\frac{1}{2}n(n-1)}\sum_{i \neq j} d(i,j)$$

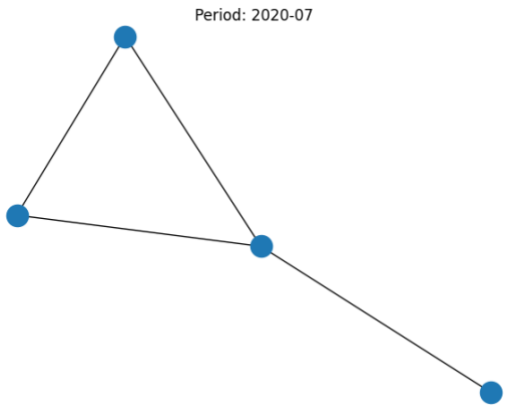
Where

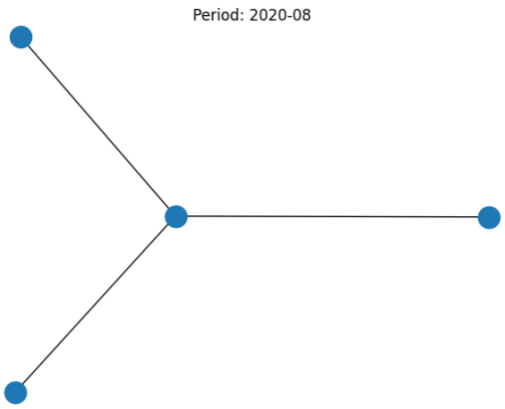
- $d(i,j)$ is the shortest path length between nodes i and j.

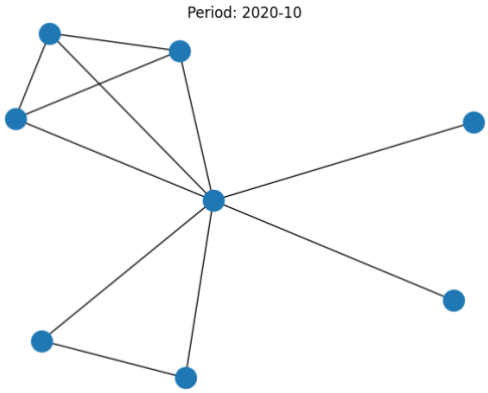
C. Case Study Repository

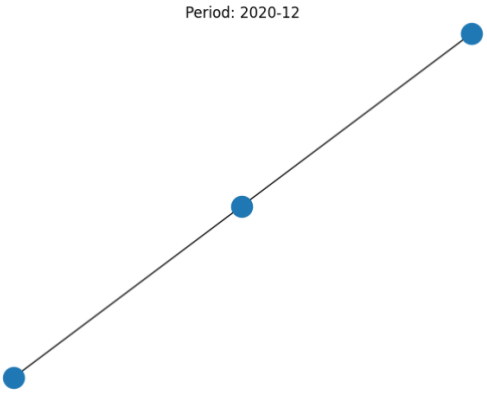
Within this section a case study is presented. This repository with the repository ID of MDEwOlJlcG9zaXRvcnkyNDY5MjkzNjI=, or called covid19-dashboard, from the company GitHub, provides a more detailed presentation of the construction of a social network. Including various monthly features for the entire network diagram for the duration of the technical preview (starting in June 2020 until the public release). Which are the constructed networks, a degree histogram, and an explanation of the network's developments.

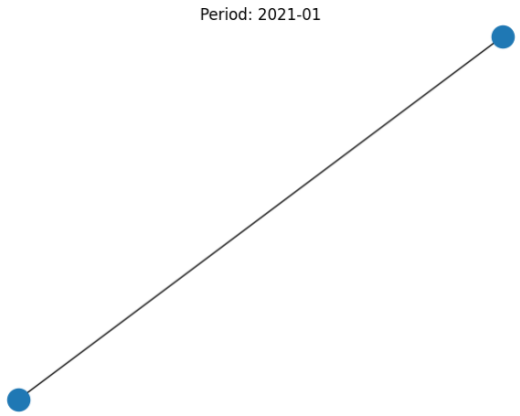



Period: 2020-07

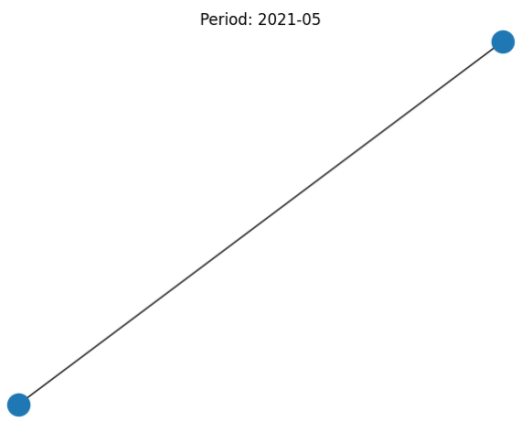

Period: 2020-08

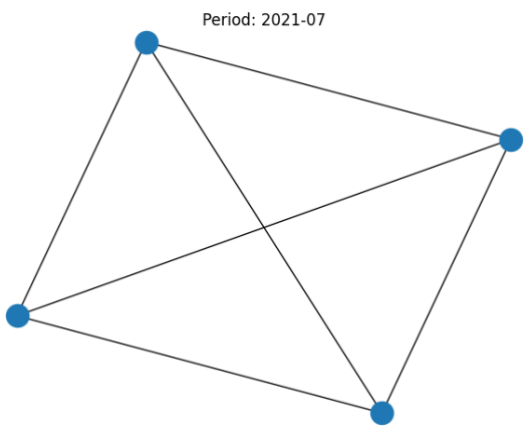

Period: 2020-10


Period: 2020-12


Period: 2021-01


Period: 2021-03


Period: 2021-05


Period: 2021-07

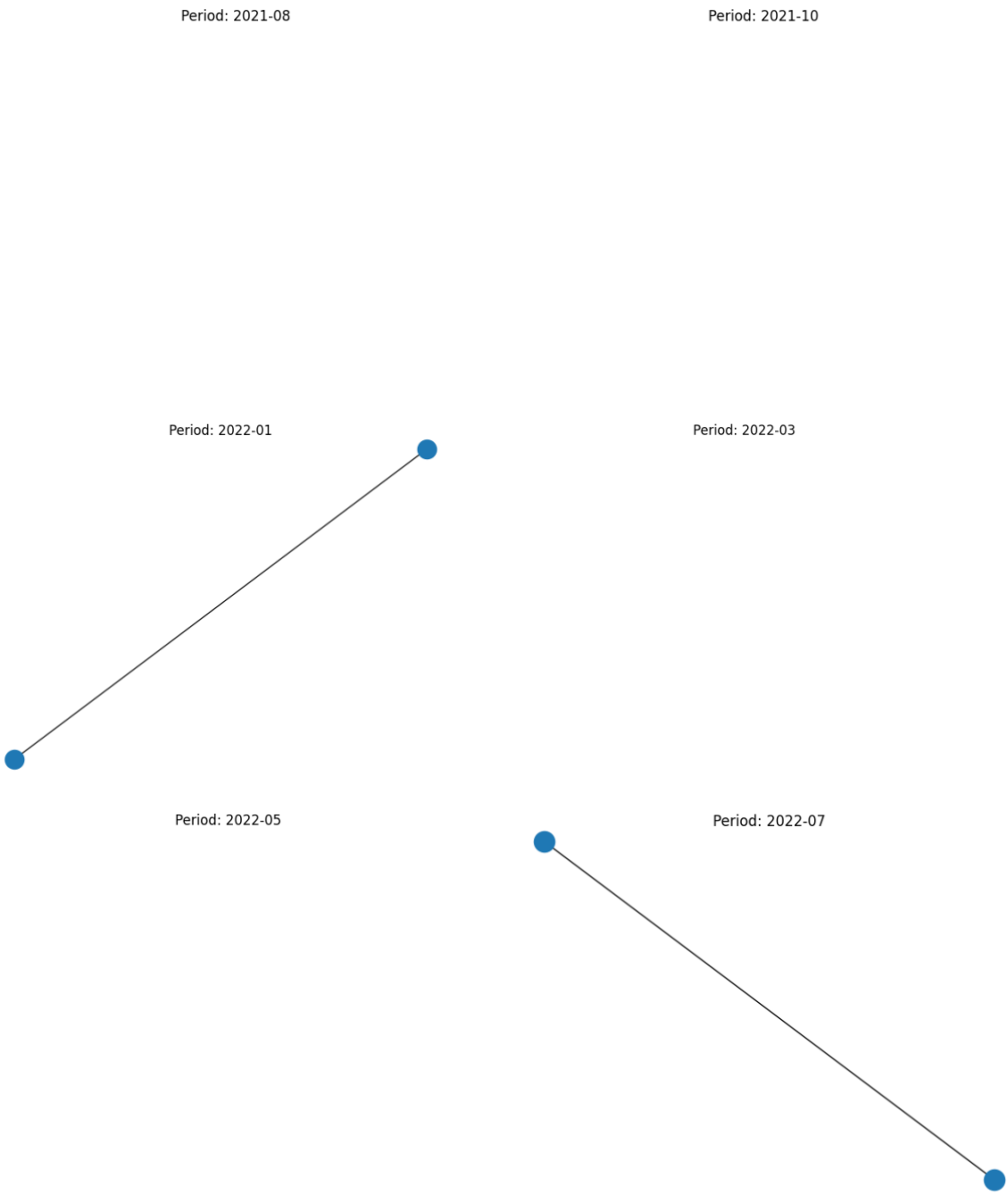


***Figure 6**: Case Study Repository Networks*

What is interesting is that the starting repository became an immediate active network in March 2020. Which make sense considering that COVID-19 arrived in the Netherlands in February 2020. Indicating an increasing need for Covid dashboard software development. Besides the active start, the visualisations present a large number of empty constructed networks. These typically would not exist if pull request data for particular month was not available. However, after further examination of the pull request data, it reveals that there is an observation here, but the participants column is empty. This situation frequently arises in the analysis.

Figure 7 also shows a degree histogram, which is a graphical representation of the degree distribution of a network. Here, year 2020 in month 3 is presented. Each bar in the histogram represents the number of nodes in that network that have certain amount of connections. The height of the bar indicates how many nodes have that many connections. And as mentioned above, this month represents an active network.

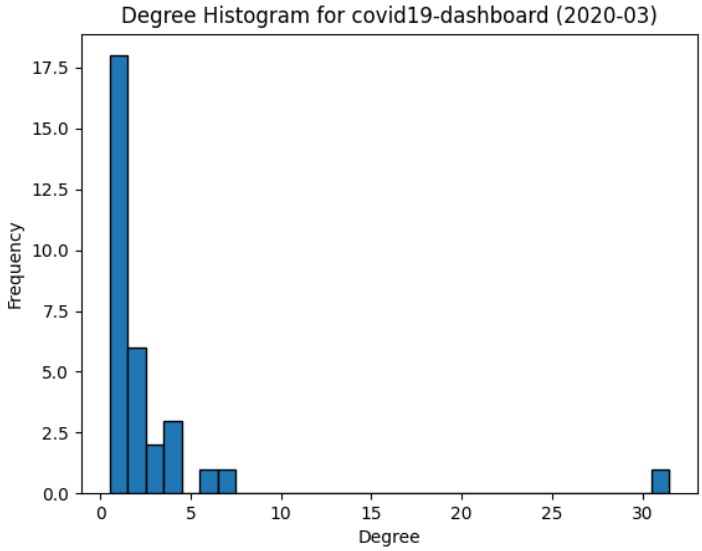


Figure 7: *Case Study Degree Histogram*