# “Mining” Ethereum: Open-Source Software Development and Community Dynamics

# Abstract

With over 152,000 developers, the open-source software () in the worldHow does such a large OSS community acquire and retain development talent to its ? This study aims to better understand Ethereum’s open-source ecosystem based on both **OSS theory** and large-scale **data analysis**. Parsing through over 19 TB of GitHub data, we construct collaboration networks among Ethereum developers, and then leverage theories in open-source software research to test several hypotheses regarding how community dynamics interact with project development. As repository levels, we find that project awareness is positively (negatively) associated with the participation of peripheral (core) developers, and has a hump-shaped association with project superposition (the average number of new releases per new commit). In addition, contribution to a project is positively associated with the project’s degree and eigenvector centrality within the developer collaboration network. These findings provide unique data-based insight into Ethereum's OSS development and community dynamics.

Key Words: Open-Source Software, Networks,

# Introduction

The past decade has witnessed stunning growth of blockchain technologies and relatedly the cryptocurrency market. Besides its technological and financial implications, the rise of blockchains as large-scale open-source software (OSS) development efforts also poses new operational and managerial questions. It is therefore timely to understand the development process and community dynamics within these OSS projects based on objective data-driven analysis.

In this paper, we tackle the above question with a large-scale data analysis of the development dynamics within Ethereum, a smart contract platform conceived by Vitalik Buterin in 2013 with an ambitious goal toward creating a decentralized “world computer.” Formally founded in late 2013 by eight “cofounders” and a small team of developers (Shin 2022), the Ethereum developers’ community has grown to over 150,000 developers in 2023 (based on the authors’ data to be discussed below).

Several features make Ethereum a desirable topic of interest for our study. First, its economic significance. As the second largest blockchain by market cap (surpassed only by Bitcoin), Ethereum’s native currency, either, is valued more than $220B in total as of 2023 summer. Furthermore, as a smart contract platform, Ethereum also supports a variety of fungible (ERC-20) or non-fungible (ERC-721) tokens, further increasing its economic significance. A wide range of applications also run on the Ethereum infrastructure, including OpenSea – one of the largest NFT marketplaces, Tether and MakerDAO – one of the largest stablecoins, Compound and Aave – the largest decentralized lending protocols, and Uniswap – one of the largest decentralized cryptocurrency exchanges. Indeed, in one quarter of 2022, the total number of transactions on the Ethereum blockchain amounted to 105.58 million generating revenue for the transaction validators of 834,874 ETH, or close to $1.68 billion (Alchemy 2022). In addition, the total value locked (TVL) in DeFi smart contracts running on the Ethereum blockchain was $89.5 billion in Q1 2022.

Second, its active development. While Bitcoin, the largest cryptocurrency, has been following a rather stable development/maturity pattern, Ethereum on the other hand follows a rather active development. For example, in August 2021 Ethereum experienced a major change to transaction fee calculations (EIP 1559), while in September 2022 Ethereum changed its consensus protocol from proof-of-work to proof-of-stake (the “Merge”). Changes like these fundamentally alter how Ethereum functions and involve significant development inputs. Such active development thus further necessitates the analyses of the open-source development dynamics as carried out by our research. Indeed, Ethereum Foundation itself has also stated that “understanding the OSS community dynamics as one of the top priorities to build a long-term foundation for credibly neutral blockchains” (Ethereum 2023).

Our data-driven analysis includes both illustrating developer networks and testing several hypotheses of their relationships with project successes. Specifically, we calculate repository statistics for all Ethereum development projects and construct collaboration networks among Ethereum developers by parsing through over 19 TB of GitHub records. We then leverage theories in open-source software research to test several hypotheses regarding how community dynamics interact with project development. As repository levels, we find that project awareness is positively (negatively) associated with the participation of peripheral (core) developers and has a hump-shaped association with project superposition (the average number of new releases per new commit). In addition, contribution to a project is positively associated with the project’s degree and eigenvector centrality within the developer collaboration network. These findings provide unique data-based insight into Ethereum's OSS development and community dynamics.

Our contributions are multifold: First, our results enrich our existing knowledge of OSS development. Cite the prior OSS literature and highlight our unique contribution – TBD

Second, our results enrich our understanding of the dynamics within decentralized systems. Although a big selling point of blockchains is in opening the *technological possibilities* of decentralization, it is an empirical question of whether decentralization can turn out to be an *economic reality*. Past research has shown that in many aspects, the blockchain ecosystem features rich dynamics that are far from a homogenous perfectly competitive market, including the mining market (Cong, He, and Li 2021, Arnosti and Weinberg 2022), validator market (Benhaim, Falk, and Tsoukalas 2021), block creation market (e.g., [MEV Boost](https://github.com/flashbots/mev-boost)), etc. Despite excellent casual discussions of the development process of blockchain projects (e.g., in Narayanan, Bonneau, Felten, Miller, and Goldfeder 2016), however, we are not aware of any systematic data-driven research into the blockchain developer community dynamics.

Finally, [need to think of one more angle – if we can; otherwise we remove this paragraph.]

# Data Collection

Our open-source software development data comes from GitHub, one of the most popular platforms among developers across the world to store and manage their codes for [software development](https://en.wikipedia.org/wiki/Software_development" \o "Software development) and [version control](https://en.wikipedia.org/wiki/Version_control" \o "Version control) using [Git](https://en.wikipedia.org/wiki/Git" \o "Git). GitHub provides over 20 event types, ranging from new commits and fork events, to opening new issue tickets, commenting, and adding members to a project (GitHub 2023). We focus on activities most commonly used by software developers to assess project quality and make decisions on whether to join the open-source project (Dabbish et al. 2012), namely commits, pull requests, issues, forks, and watches. Specifically, according to [GitHub Glossary](https://docs.github.com/en/get-started/quickstart/github-glossary" \l "issue), a **commit**, or "revision", is an individual change to a file (or set of files);[[1]](#footnote-1) A **pull request** is a proposed change to a repository (akin to a folder) submitted by a user, while an **issue** is a suggested improvement, task or question related to the repository. Pull requests and issues can be created by anyone (for public repositories) and are moderated by repository collaborators (who may accept or reject a pull request);[[2]](#footnote-2) A **fork** is a user’s personal copy of another user’s repository that lives on the first user’s account, which allows one to freely make changes to a project without affecting the original upstream repository;[[3]](#footnote-3) A **watch** allows a user to follow a repository or issue to receive notifications when updates are made to an issue or pull request.

In this study, we focus on the OSS development activities within Ethereum, which lives under Ethereum’s main webpage on GitHub: <https://github.com/ethereum>. The structure of the GitHub platform allows the creation of different repositories under the main Ethereum webpage, and all these repositories share a common pattern of addresses that begin with https://github.com/ethereum/<repository name>. Therefore, we extract all activities within repositories whose addresses start with this pattern to filter Ethereum development data from GitHub. Since GitHub allows users to “fork” other repositories, these forked repositories will see patterns like “<user webpage>/ethereum/” in their repository addresses. We do not collect data for activities within such forks. We collect data from December 2013, when Ethereum first start development, to May 2023 and aggregate all activities at the repository - user (i.e., actor.login) - day level.

To gather all these GitHub activities, we take advantage of GitHub Archive hosted on the Google BigQuery platform (Hoffa 2016), which comprises the historical data and content of more than 2.8 million open-source GitHub repositories. The data is stored in a hybrid database format that combines structured and unstructured data. Google BigQuery platform infrastructure allows SQL-like queries to access the entire dataset and other non-SQL requests (BigQuery 2023). For each event (e.g., commits, pull requests, issues, forks, and watches) we are interested in, GitHub Archive also allows us to extract more details from a payload string, which field contains the JSON encoded activity description with detailed information about the content of the changes made, previous version of the file, comments developers have attached to pull requests or commits, and whether a pull request or issue were closed.

GitHub Archive began to collect data in 2011. However, a change in API altered the data structure and created a discrepancy between record schemas before and after 2015 (specifically, development activity records on dates between 2/12/2011-12/31/2014 were recorded from the Timeline API, which was subsequently replaced by the Events API starting from 1/1/2015). To alleviate such discrepancy and unify the data structure for the proposed analysis, we extracted the missing information from the payload fields for records before 2015. Since Ethereum’s OSS development started in late 2013, over one year of data has been extracted using the alternated query and processed to match the rest of the dataset.

Overall, we collect 460,000 records after processing 19.08 TB of data. Using the initial dataset, we further aggregate the data at a repository-day level to test some of our hypotheses. Table xxxx provides summary statistics of our collected data.

Insert the summary statistics we had before here.

# Developer Network

Insert the network illustrations we had before here.

A network of orange dots and lines

Description automatically generated

# Hypothesis Testing

Using our data, we further test several hypotheses inspired by the existing OSS literature.

## Hypothesis 1

Our first hypothesis concerns the association of repository popularity with the involvement of different types of developers. Specifically, developers who contribute to a repository can be classified into two types: core and peripheral. A core developer is one who is heavily involved in the project development, typically has the right to apply proposed changes to the main software file, or merge pull requests, and strategically decide the future functionality of the software product. Peripheral developers are contributors who are not core developers. Based on the literature, core developer has been defined as contributing more than 12% of code to the project (Mockus et al. 2002). We adopted this threshold and marked as core developers those who contributed 12% or more of total activities in a repository over the past 30 days. All other developers were marked as peripheral. We also checked for the abnormal patterns of contributions by unique contributors, e.g., one across all repositories on a daily basis, and ran our analysis without them. Typically, such activity is associated with bots and should not be considered in our analysis. Moreover, the login name of such actors contains the ‘bot’ part, and we mark them as ‘is\_bot’ in our dataset. The results with and without bot activities are robust.

H1: The roles of core versus peripheral developers: *Greater participation of peripheral developers in open-source projects is associated with greater awareness about open-source products* (Setia et al. 2012).

To test our first hypothesis, we run a panel fixed effects regression:

(1)

In equation (1), is the index of a repository within the Ethereum project, is the index of a time period, is the repository fixed effect that accounts for unique attributes of the repository that are not captured by other variables, is an idiosyncratic error. The main independent variable is the number of peripheral developers (Peripheral). We also control for the number of core developers (Core) and the lifetime of the project since its initiation (Duration).

For the left-hand side variable, as a measure of product awareness (), we use the number of repository forks (Forks) and the number of watches (Watches), both of which have been quoted in the previous literature to measure project awareness. Developers tend to fork the repositories they are interested in for two reasons: (1) they want to contribute to the development and improvement of the software, and (2) they want to use the existing code for the development of their own project. In both cases the motivation to fork is rooted in the inherent quality of the software and its recognition by developers. In addition, the technology diffusion happens by means of forks, and the wider population becomes aware of it. Watching a repository is similar to following the account on the social media as it enables to receive the notifications about any changes and updates applied to the watched account. Therefore, the number of watches on GitHub is the direct measure of the popularity of the software project.

Our regression results confirm Hypothesis 1: The number of peripheral developers is positively associated with the repository awareness (number of forks and number of watches). The addition of one peripheral developer increases the number of repository forks by 0.16 and number of watches by 0.52 on a daily basis. The results are presented in Table 1.

**Table 1. H1 Testing Results**

|  |  |  |
| --- | --- | --- |
| Variable | (1)  FE, DV=Forks | (2)  FE, DV=Watches |
| Peripheral | 0.1623\*\*\* | 0.515\*\*\* |
|  | (0.0027) | (0.0071) |
| Core | -0.0042\*\* | -0.055\*\*\* |
|  | (0.0016) | (0.0107) |
| Duration | 0.0003\*\*\* | 0.0001\*\* |
|  | (0.00002) | (0.00002) |
| Repository FE | Y | Y |
| Num obs | 101,081 | 101,081 |
| Adj R-sq | 0.646 | 0.833 |

**Note.** Standard errors in parentheses; + p<0.1, \* p<0.05, \*\* p<0.01, \*\*\* p<0.001

The result in Table 1 shows that the awareness of a repository increases when more developers show interest in it but are not yet heavily involved in its development activity (less than 12% of all activities in the past 30 days). A likely reason is that peripheral developers are involved in many other projects on the platform, therefore, when interaction with the focal repository is reflected on their profile, their collaborators are exposed to the focal project. The attention to the focal project can be reflected by subscribing for the repository updates (watching) or forking the code for future modification or use. While watching serves as a powerful signal of popularity of the code, number of forks signify the social recognition and impact (Petryk et al. 2023).

## Hypothesis 2

Due to the importance of the social connections in the OSS community for the information diffusion, we further investigate their impact with the next hypotheses.

H2: The nuanced relationship between developer repeated collaboration and project success: A moderate level of internal cohesion within a project is better for a project’s success than very high or very low levels of internal cohesion (Singh et al. 2011).

To test our second hypothesis, we construct the measure of internal cohesion following Singh et al. (2011). The presence of repeated collaborations among project developers is related to strong interpersonal connections (Uzzi 1997). We calculate the number of developer pairs from the focal repository that worked on other repositories within Ethereum project and divide it over the total number of pairs that exist in the focal repository to calculate the number of repeated ties, or the internal cohesion metric. The resulting metric is the main independent variable. The main dependent variable is a project’s success. Subsequent to the extant literature, we consider a project’s successas a project’s rate of knowledge creation and measure it as number of commits (Boh et al. 2007, Crowston et al. 2003). We also control for the the lifetime of the project since its initiation (Duration).

To test our second hypothesis, we run a panel fixed effects regression:

(2)

In equation (2), i – is the index of a repository within the Ethereum project, t – index of a time period, is the repository fixed effect that accounts for unique attributes of the repository that are not captured by other variables, – idiosyncratic error. include the lifetime of the project since its initiation ().

The results are presented in Table 2.

**Table 2. H2 Testing Results**

|  |  |  |  |
| --- | --- | --- | --- |
| Variable | (1)  OLS, DV=lnCommits | (2)  RE, DV= lnCommits | (3)  FE, DV= lnCommits |
| mcIntCohesion | -0.002\*\* | -0.004 | -0.004 |
|  | (0.002) | (0.005) | (0.005) |
| mcIntCohesion^2 | -0.000004 | 0.0001 | 0.0001 |
|  | (0.00001) | (0.00006) | (0.0001) |
| Duration | -0.0001\*\*\* | -0.0001\*\*\* | -0.0001\*\*\* |
|  | (0.000003) | (0.00004) | (0.00004) |
| Intercept | 0.660\*\*\* | 0.452\*\*\* |  |
|  | (0.004) | (0.026) |  |
| Repository FE | N | N | Y |
| Num obs | 101,080 | 101,080 | 101,080 |
| Adj R-sq | 0.014 | 0.016 | 0.011 |

**Note.** Heteroskedasticity-consistent and autocorrelation-corrected standard errors in parentheses; + p<0.1, \* p<0.05, \*\* p<0.01, \*\*\* p<0.001

## Hypothesis 3

Building on the prior evidence of the relationship between project popularity and structure of contribution, we point our attention to the study by Medappa and Srivastava (2019) that suggests the nonlinear relationship between the superposition of the project and its popularity. We follow with the third hypothesis:

H3: The nuanced relationship between project structure and popularity: *The degree of superposition, that is, the ratio between the total number of versions of the OSS project to the total number of individual contributions to the project, has a nonlinear relationship with project attractiveness in the OSS community: A moderate degree of project superposition is better for a project’s popularity than very high or very low levels of superposition* (Medappa and Srivastava 2019).

To test our third hypothesis, we construct the measure of superposition as the main independent variable. We calculate the number of developers who contribute to the repository over a time period and divide it over a number of releases made within the same time frame. The resulting metric is distributed between 0 and 1 by definition. If degree of superposition equals exactly 1, all of the releases were implemented by individual developers and added sequentially. The degree of superposition decreases as a project adopts a concurrent development approach and approaches 0 as a greater number of individual contributions get accumulated into individual releases (versions) of the project. Since our dependent variable is *project’s popularity* and is congruent with the dependent variable in our first hypothesis, we also control for the number of core and peripheral developers in our analysis.

We run a panel fixed effects regression (3) to test our third hypothesis:

(3)

In equation (3), i – is the index of a repository within the Ethereum project, t – index of a time period, is the repository fixed effect that accounts for unique attributes of the repository that are not captured by other variables, – idiosyncratic error. include the number of peripheral developers (), the number of core developers (), and the lifetime of the project since its initiation ().

The results are presented in Table 3.

**Table 3. H3 Testing Results**

|  |  |  |
| --- | --- | --- |
| Variable | (1)  FE, DV= Watches | (2)  FE, DV=Forks |
| Superposition | 14.967\*\*\* | 6.412\* |
|  | (3.727) | (2.757) |
| Superposition^2 | -19.145\*\*\* | -7.903\* |
|  | (5.790) | (3.550) |
| Peripheral | 0.487\*\*\* | 0.206\*\*\* |
|  | (0.053) | (0.024) |
| Core | 0.002 | 0.002 |
|  | (0.005) | (0.002) |
| Duration | -0.0004 | 0.0001 |
|  | (0.00004) | (0.0002) |
| Repository FE | Y | Y |
| Num obs | 1,593 | 1,593 |
| Adj R-sq | 0.756 | 0.645 |

**Note.** Heteroskedasticity-consistent and autocorrelation-corrected standard errors in parentheses; + p<0.1, \* p<0.05, \*\* p<0.01, \*\*\* p<0.001

The result in Table 3 signifies that there is a non-linear relationship between superposition and project popularity ( is significant). As the level of superposition in the project increases, the project popularity increases but to a certain point. After a certain point, the further increase of superposition the project popularity starts decreasing ( is negative). Low superposition signifies that project releases occur after large number of contributions are accumulated. Such “productive deferral” makes the project less attractive to the developers due to lack of autonomy and independence in completing complex tasks. Hence, the developers are reluctant to follow such projects (Ryan and Deci 2000). High superposition signifies the high level of work independence that satisfies the autonomy need. However, the lack of collaboration and exchange of ideas creates the negative affective state and deters the interest to the project (Medappa and Srivastava 2019). Developers need the decent amount of challenge and autonomy to be interested in contributing to the project and opportunities to work with other developers on more complex tasks (Ke and Zhang 2010).

## Hypothesis 4

Due to the importance of the social connections in the OSS community for the information diffusion, we further investigate their impact with the next hypotheses.

H4: The importance of a project’s embeddedness among other OSS projects: *A project’s visibility and embeddedness in the global OSS community is positively associated with the project success.* (Grewal et al. 2006)

To test our fourth hypothesis, we construct the network of Ethereum repositories where the edges are established by the common collaborators. We use the R package igraph to do the computations. Based on the network, we calculate the centrality measures: degree centrality (), betweenness (), and eigenvector centrality () – to include as the main independent variables in our regression. We measure *project’s success* as the number of commits (Grewal et al. 2006) and number of forks and include them as the main dependent variables in our analysis. We also control for the lifetime of the project.

We run a panel fixed effects regression (4) to test our fourth hypothesis:

(4)

In equation (4), *i* – is the index of a repository within the Ethereum project, is the constant term that accounts for factors that affect the project success metrics not captured by other variables, – idiosyncratic error. As a measure of product success (), we use a metric of contribution – a number of commits (Commits) and a metric of diffusion – a number of repository forks (Forks). are the coefficients of interest and measure the effects of centrality measures on the project technical success. include the lifetime of the project since its initiation ().

The results are presented in Table 4.

**Table 4. H4 Testing Results**

|  |  |  |
| --- | --- | --- |
| Variable | (1)  OLS, DV= lnCommits | (2)  OLS, DV= lnForks |
| Degree | 0.011\*\*\* | 0.007\*\*\* |
|  | (0.002) | (0.001) |
| Eigenvector | 9.112\*\*\* | 7.094\*\*\* |
|  | (2.238) | (1.391) |
| Betweenness | -0.001\*\*\* | -0.001\*\* |
|  | (0.0004) | (0.0002) |
| Duration | -0.0004\*\* | 0.001\*\*\* |
|  | (0.0002) | (0.0001) |
| Intercept | 0.769\*\* | 0.098 |
|  | (0.310) | (0.161) |
| Num obs | 400 | 400 |
| Adj R-sq | 0.319 | 0.713 |

**Note.** Heteroskedasticity-consistent and autocorrelation-corrected standard errors in parentheses; + p<0.1, \* p<0.05, \*\* p<0.01, \*\*\* p<0.001

The result in Table 4 signifies that there is a statistically significant relationship between a network structure and project success. As the degree centrality increases, the projects become more interconnected with others by the common developers that attracts more contributions ( in column 1) and the technological innovation produced in the focal project is being reused by many other developers ( in column 2). As the eigenvector centrality increases, the projects become more central in the network that turns them into magnets with respect to more contributions ( in column 1) and the diffusion of innovation ( in column 2). Lastly, As the betweenness centrality increases, the flow of talent becomes easier due to the project’s connectivity to other repositories that leads to reduction in contributions to the focal project ( in column 1) and the project forks ( in column 2).

# Discussion

TBD

Future work: other Ethereum related projects, network dynamics, other crypto projects (ecosystems), causal inferences.

# Reference List

1. Hoffa F. 2016. GitHub on BigQuery: Analyze all the open source code. <https://cloud.google.com/blog/topics/public-datasets/github-on-bigquery-analyze-all-the-open-source-code>
2. BigQuery 2023. <https://cloud.google.com/bigquery/>
3. GitHub 2023. <https://docs.github.com/en/webhooks-and-events/webhooks/webhook-events-and-payloads>
4. Mockus, A., R. T. Fielding, J. D. Herbsleb. 2002. Two case studies ofopen source software development: Apache and Mozilla.ACMTrans. Software Engrg. Methodology11(3) 309–346.
5. Petryk, M., Qiu, L., and Pathak, P. 2023. The Impact of Open-Source Community on Cryptocurrency Market Price: An Empirical Investigation. MISQ, Forthcoming.
6. Uzzi, В. 1997. "Social Structure and Competition in Interfirm Networks: The Problem of Embeddedness," Administrative Science Quarterly (42:1), pp. 35-67.
7. Ke W, Zhang P (2010) The effects of extrinsic motivations and satisfaction in open-source software development. J. Assoc. Inform. Systems 11(12):784–808.
8. Ryan RM, Deci EL (2000) Self-determination theory and the facilitation of intrinsic motivation, social development, and well-being. Amer. Psych. 55(February):68–78.
9. Crowston, K., H. Annabi, J. Howison. 2003. Defining open source software project success. Proc. Internat. Conf. Inform. Systems, ICIS, Seattle, WA, Association for Information Systems, Atlanta, GA.
10. Boh, W., Slaughter, S., and Espinosa, J. 2007. "Learning from Experience in Software Development: A Multilevel Analysis," Management Science (53:8), pp. 1
11. Shin, Laura. The Cryptopians: Idealism, Greed, Lies, and the Making of the First Big Cryptocurrency Craze. PublicAffairs, 2022.
12. Cong, Lin William, Zhiguo He, and Jiasun Li. "Decentralized mining in centralized pools." The Review of Financial Studies 34, no. 3 (2021): 1191-1235.
13. Arnosti, Nick, and S. Matthew Weinberg. "Bitcoin: A natural oligopoly." Management Science 68, no. 7 (2022): 4755-4771.
14. Benhaim, Alon, Brett Hemenway Falk, and Gerry Tsoukalas. "Scaling blockchains: can committee-based consensus help?." arXiv preprint arXiv:2110.08673 (2021).
15. Narayanan, Arvind, Joseph Bonneau, Edward Felten, Andrew Miller, and Steven Goldfeder. Bitcoin and cryptocurrency technologies: a comprehensive introduction. Princeton University Press, 2016.







1. When a developer makes a commit to save work, Git creates a unique ID (a.k.a. the "SHA" or "hash") that allows him/her to keep record of the specific changes committed along with who made them and when. Commits usually contain a commit message which is a brief description of what changes were made. [↑](#footnote-ref-1)
2. Both pull requests and issues may have their own discussion forums. [↑](#footnote-ref-2)
3. One can also open a pull request in the upstream repository to keep the fork synced with the latest changes since both repositories are still connected. [↑](#footnote-ref-3)