Identifying Developer Engagement in Open Source Software Blockchain Projects through Factor Analysis

Jeff Nijsse
Auckland University of Technology
jeff.nijsse@aut.ac.nz

Alan Litchfield
Auckland University of Technology
alan.litchfield@aut.ac.nz

Abstract

The ubiquity of GitHub for software developers to coordinate software development in a community platform has resulted in a rich source of public data. Blockchain teams put open source code as a founding principle since the release of Bitcoin and nearly all blockchain-based projects have code visible on GitHub. Developer engagement is known to be important to the health and viability of open source software, yet has varying definitions and no standard method of measuring what constitutes engagement.

This work uses exploratory factor analysis to identify dimensions that represent engagement in a community of open source developers. We find that a latent factor composed of pull-requests, commits, comments, and authors based on a monthly average of the previous three months is a representation of Developer Engagement. A secondary factor consists of stars, forks, and total authors. Cross validation of the dataset is carried out with good support for the model.

1. Introduction

GitHub represents the largest open source software (OSS) community hosting over 254 million repositories by 73 million developers [1, 2, 3]. Since the rise of Bitcoin, cryptocurrencies and their derivative blockchain projects have grown dramatically and the present study finds that 98.6% of the top 419 public repositories are hosted on GitHub. This predominance has emerged from an open source ethos that includes decentralization and today represents the de facto standard for blockchain projects, contributing to the speed of innovation, iteration (and failure). The ease of access to version control data in OSS has allowed for fine-grained and access, collection, and analysis of data within software communities. Most of the academic focus has been at an individual level within large projects, with less concentration on the project-level. Within these studies, developer engagement is known to be important [4, 5, 6] but does not itself have a clear construct of the components that make up engagement. Software community engagement is a subjective term referring to how people interact and contribute to a project often without financial incentive [7]. Engagement in social media is often measured by likes or re-posts or comments; these have parallel instances in GitHub which has been referred to as a social coding platform [2, 3].

The goal of this work is to build usable knowledge of engagement within OSS blockchain projects through the use of public OSS data from GitHub. We hypothesize that publicly available data can be used to identify measurable factors that influence OSS developer engagement.To do this, the study draws from the myriad event types that are logged by GitHub to identify factors that contribute to OSS engagement.

This work represents part of a larger study investigating health of OSS blockchain projects and implications for innovation. Exploratory factor analysis (EFA) is used here as part of the discovery process for a structural model. Once indicators of engagement have been identified, researchers and practitioners can pinpoint areas of strength and weakness. They can also build more comprehensive statistical models.

1.1. Open Source Software

We utilize the definition of open source by Bruce Perens [8], that OSS is free to distribute without royalties, has published source code, and can be modified by anyone with few conditions. In the context of this study there must be source code and version tracking information available for analysis. It is noted that there are different types of open source licenses, and this is not of concern to the present work. We do not make any assumptions or exclusions based on a projects' software license. The canonical OSS definition can be found online[1] including an overview of the types of licenses[2] developers can choose from when creating software.

[1] https://opensource.org/osd.html
[2] https://opensource.org/licenses

URI: https://hdl.handle.net/10125/103285
978-0-9981331-6-4

The main contributions of this article are twofold. First, we create a clean and complete dataset of blockchain projects' source code metrics which is made publicly available. Second, we conduct an empirical study using factor analysis to identify the contributing variables that make up an intangible dimension of OSS engineering: *developer engagement*.

2. Related Work

Very few multivariate statistical approaches such as factor analysis and structural equation modelling (SEM) have been applied to open source software engineering. More work has been done in the area of individual developer engagement (see Section 4.2). To our knowledge this is the first work applying factor analysis to OSS projects in the blockchain space investigating engagement at a project level.

In the context of software health and success, Poba-Nzaou & Uwizeyemungu's study on worry profiles of OSS developers uses factor analysis on worry profiles of OSS contributors [4]. Focusing on the individual developer they identify twelve challenges to OSS development, nine of which can be related to engagement. Abdulhassan [9] modelled trends in GitHub programming languages via SEM, using stars to identify the most popular repositories. Raja & Tretter [10] develop a regression model for software viability (survivability) that consists of three elements: version releases, response time to artifact requests, and average mutual information.

Wang et al [11] identify activities undertaken by developers in addition to standard coding including: managing the community, parallel development, documentation, and participating in discussions. Their study is centered around the individual developer whereas this work is interested in total engagement at the project-level. Standard coding-type activities are captured by GitHub event types and discussed in Section 4.2.

Tamburri et al. [12] include engagement in a list of six key characteristics of OSS. The remaining five are: community structure, geodispersion, longevity, formality, and cohesion. Fang & Neufeld [5] studied the effects of sustained developer participation (engagement) within a large OSS organization through the lens of Legitimate Peripheral Participation theory. Shaikh & Levina [6] evaluated an open source communities' predisposition to corporate partnership and found three indicators of viability: vibrancy (number of contributors, growth, and opportunity for new members), growth of code base, and quality that contribute to a healthy ecosystem. Borges et al. [13] study the factors that affect popularity in GitHub. They use stargazers as a proxy for popularity and conclude that programming language, domain, and repository ownership classification (user/organization) all affect the number of stars.

3. Methodology

The research methodology details the rationale for applying exploratory factor analysis in Section 3.1 to look for indicators that represent developer engagement. The objective of this method (Section 3.2) is to summarize the data into succinct dimensions, and then Section 3.3 answers the type and number of variables to be included.

3.1. Exploratory Factor Analysis

Exploratory factor analysis (EFA) is a multivariate statistical technique used for determining underlying constructs that are present in a dataset composed of a large number of variables. The constructs, or factors, can represent groupings within the data that are hypothesized, or known, but not directly observable. This unveils structure within the dataset and is often called Factor Analysis outside of the social sciences [14]. *Latent variables* and *factors* are terms used interchangeably and represent inherent characteristics that are difficult to measure. EFA is common in fields like psychology and economics where participants are given a survey and the results are analysed [15] but little work has been done applying the technique to software engineering. OSS is a human lead directive combining social coordination with technical innovation, and as a socio-technical field is fit for application of these techniques to capture inherent structure such as 'engagement'.

EFA differs in theory and practice from principle component analysis (PCA). PCA sets the value of the correlation matrix diagonal to unity to consider all the variance including self-correlation [16]. The main goal of a PCA is to reduce the number of variables and parsimoniously capture signal. When finalising a PCA the result is that the measured variables influence the derived components, for example, within the present context: "more commits positively influence developer engagement." EFA assumes that the latent construct is responsible for the correlation and so separates the communality (shared variance, u^2) from the unique variance (h^2). EFA cannot account for the difference between specific variance (s^2) and error variance (e) and these are thus grouped with communality such that $h^2 = s^2 + e$ [17]. The diagonal of the correlation matrix shows the estimate of the common variance that is accounted for. In practice this reverses the statement of influence to say: "engagement is positively related to commits." This method does not assume perfect measurement of ob-

served variables and allows the factor to explain what the indicators have in common, and what is not held in common is due to measurement error [14].

The EFA approach is more applicable here because "the goal of research is to identify latent constructs for theory building" [18]. Additionally, EFA allows for correlation between latent variables which is to be expected in the confirmatory factor analysis stage of SEM. In this study EFA is used in the discovery stages of a more-detailed confirmatory factor analysis. We follow the six stages of EFA in the framework by Hair [16] beginning with the research objectives.

3.2. Objectives of the EFA

The first stage is to identify the research problem which in this case is using open source data to search for underlying factors, or dimensions, that are not directly observable; developer engagement. The research is taking an exploratory approach (not confirmatory), as part of the discovery process for a structural equation modelling study. The goal here is to achieve data summarization rather than reduction as there are few indicator variables to assess. The set of OSS projects is limited to blockchain-based software to reduce the influence from obvious data clusters as adjacent fields such as mobile development may have different foundational principles.

3.3. Research Design

The research design stage answers how many, and what type of variables are to be included? Metrics are determined to adequately capture the characteristics of developer engagement, then data is collected and cleaned. Once the metrics have been calculated, the dataset can be analysed. These steps are common in extracting and analyzing open source data [19].

We have selected eight variables that can be determined and calculated from open source data. All eight can be considered as contributing to the hypothesized factor of engagement as a proxy for cumulative developer contribution and activity. All variable data is metric and ratio (exists on a linear scale, e.g., 4 months is twice as long as 2 months) and can be calculated from the dataset derived from the archive of GitHub events.

Best practices for factor derivation are to have three indicators per dimension [17, 18] and for exploratory purposes at least five [16] to better assist with structure identification. A single factor consisting of four variables is considered over-identified; three variables is just-identified; and two variables is under-identified. It is best to avoid situations of under-identified factors. The goal of this study is to identify at least one factor beginning with eight indicators present.

The complete dataset used in this study consists of 393 GitHub repositories, each from a different project. The desired sample size is a minimum of 200 as researchers propose in the range from a minimum of 5 observations per variable to as many as 20 per variable [16]. More observations are used because of the higher requirements for structural equation modelling (future work), especially under non-normal data conditions. This also assists with meeting sample size requirements for cross-validation.

While the key assumptions made at this point involve conceptual decisions, statistical assumptions are discussed in Section 4.3. Data homogeneity is ensured by selecting a subset industry of the GitHub ecosystem. The selection must be broad enough to meet the required sample size, yet narrow enough for resource constraints. Blockchain-based projects including blockchains, cryptocurrencies, decentralized exchanges, tools, layers, bridges, etc. are selected. Other industries–web/libraries/mobile/etc.–that have OSS projects are excluded to limit the domain and potential correlation corruptions to the factor structure that subsamples could influence.

Once the data are collected (Section 4), and the factors determined (Section 5), the validation stage is carried out through a process of cross-validation (Section 5.3).

4. Data Collection

Data for public blockchain projects are readily available for collection and analysis from GitHub through the web interface, programmatically through the application programming interface (API), and in raw archival form from the GitHub Archive.

All GitHub data from February 2011 has been archived in JSON Lines format (JSON object on every line) and is available for public download from the GHArchive[3] project. Every JSON object contains the metadata and data for one GitHub event. For example, when a repository is starred an event is emitted of `type: stargazer`. Similarly an event is created when a pull request (PR) is created, and the JSON contains all the details about who created it, when it was created, and the contents of the PR object.

GitHub currently has 17 event types[4], seven of which are relevant to this study: `WatchEvent`, `ForkEvent`, `PushEvent`, `PullRequestEvent`, `IssueCommentEvent`, `CommitCommentEvent`, and `PullRequestReviewCommentEvent`. All the

[3] https://www.gharchive.org/

[4] https://docs.github.com/en/developers/webhooks-and-events/events/github-event-types

events have metadata with *author username* and *date* which can be used for further metrics.

GHArchive data was downloaded from February-2011 up to 26-March-2022, consisting of 2.36 TiB in total information. The compressed JSON is then inserted into a single-table ClickHouse database [20]. ClickHouse[5] is an open source column-oriented database management system designed for online analytical processing. This is ideal for large datasets that involve mostly read-only queries and batch updating.

The database contains 5.6 billion records and is 430 GiB and is accessed with structured query language (SQL) queries through a command line or a python module. This is running on a dedicated Linux Ubuntu (version 20.04.4) machine with ClickHouse's command line client and server (versions 22.3.3.44) installed.

The top 600 blockchain projects are identified using the CoinMarketCap[6] API by ranking of market capitalisation as of March-2022. Details retrieved include project *name*, *rank*, *website*, and *location of source code* if available. The data is collated into a Pandas dataframe (version 1.4.2) for Python (version 3.8.10) via JupyterLab (version 3.3.3).

4.1. Data Processing

In many cases the source code location is incorrectly reported and so all project source code locations are manually verified. Where the location points to an organization on GitHub, the repository (repo) with the *reference*, or *core*, or *node* implementation is chosen. If this is not the case (perhaps it is not a blockchain), then the *contract* repository is chosen. To disambiguate between competing repos, the one with the most stars is chosen. Often the core repo is also the one with the most stars. When there are two implementations in different languages (e.g., GO and Rust) highest stars takes preference. Forked libraries are not considered.

As an example of verification, the Sythetix ecosystem has six pinned repositories and is listed by CoinMarketCap as having code at `https://github.com/Synthetixio` however this is the organization landing page and contains links to all repos. The main platform is hosted at `https://github.com/Synthetixio/synthetix`, which is manually verified.

The top 600 blockchain projects are the starting point and the dataset is cleaned with the following exclusions:

- The version control data cannot be accessed for analysis. Eight have a repo that's missing (404 error) indicating it has been deleted or moved; 78 are listed but private; 83 do not have a repo listed (and are likely private).
- Four are hosted on GitLab[7], and two on Bitbucket[8]. These GitLab and Bitbucket ones are excluded because they are a small percentage of the whole (1.4%) and would require separate infrastructure to access the code bases.
- Six are doubles where the project points to the same code base for a related project, e.g. KavaSwap (SWP) and Kava (KAVA); only the highest ranked project is included.

Of the 419 publicly available repos on GitHub, 26 of these have no contribution history indicating the repo was created or the code was copied there and never updated. These are excluded as they are not representative of developer engagement in OSS.

There is no missing or null data to handle as the absence of a metric is scored with a zero. For example if there were no comments posted in the previous three months the query would return a zero. The data has been examined for influential observations (outliers) and although there are extreme values, these observations represent actual blockchain projects and are included in the spirit of producing a representative model. Exclusion of potential outliers is tricky, and without good cause such as impossible values or missing data it is best to include them [21]. Further, the EFA analysis has been completed with and without some of the possible influential observations according to the Mahalanobis distance to no marked difference in results [22]. This is good evidence to include all data.

The final dataset is available at `https://github.com/millecodex/maui` and contains full data for 393 blockchain projects on which this analysis is based.

4.2. Metrics

A broad definition of engagement is "average hours per week worked on OSS projects" by developers [4, 23], and if they interact with GitHub during that work time, then any events registered by the version control software are seen as contributing to engagement. GitHub has seventeen event-types that get logged, and these can further be combined using metadata such as *username* and the *timestamp* to determine indicators of engagement.

Many 'engagement-type' activities that developers and software communities participate in can map to GitHub events. The straight-forward ones are traditional developer activities such as improving quality, fix-

[5] `https://clickhouse.com/`
[6] `https://coinmarketcap.com/`
[7] `https://gitlab.com/`
[8] `https://bitbucket.org/`

Table 1. Descriptive statistics

$n = 393$	Mean	Std. Dev	Min	Q1	Median	Q3	Max
pull requests	14.1	37.2	0	0	2.0	13.7	524.3
comments	64.4	205.5	0	0	1.3	33.3	2440.7
authors	8.4	18.4	0	1	2.7	8.3	174.0
commits	92.2	205.4	0	0	8.7	81.7	1522.0
contrib. total	135.8	543.1	1	8	24	76	7465
days inactive	50.8	156.9	0.003	0.6	2.4	19.9	1314.6
forks	528.0	3398.5	0	12	59	204	59013
stars	772.8	4350.1	0	22	101	391	72112

ing bugs, and implementing new features, and are captured by the *push*, *pull request*, and *comment* activities [6, 24, 25, 26]. A number of studies use *author* information to derive metrics [12, 10, 6]; communication and discussion information through *comments* [12, 27]; *issues* to calculate response time and as a parallel to average mutual information [10]; and bookmarking of projects of interest through *forks* and *stars* [23]. Tamburri et al [12] use *authors*, *pull request comments*, *commit comments*, and monthly active members, among seven total indicators to represent engagement.

Building on these studies, eight metrics have been selected that meet the criteria of having a public repository hosted on GitHub and being potentially indicative of OSS engagement at a developer level.

Choosing a time-frame for certain metrics is important as projects grow and develop. Not all activity from the past is indicative of present performance. Using the recent three months helps identify active projects and avoids the peril of dead projects [28]. Conversely, shorter time-frames such as one-month can be too limited and fail to capture development milestones in progress [11].

Using the known location of the source code, a script is then written to query the database and calculate the following metrics. The descriptive statistics are shown in Table 1.

1. *Stars*—a `WatchEvent`; the total number of stars received since creation.
2. *Forks*—the total number of times a repository has been forked since creation; a `ForkEvent`.
3. *Commits*—a `PushEvent`; the total number of commits to the codebase for the previous three months, calculated as a monthly average.
4. *Pull Requests*—the number of `PullRequest Event` for the previous three months, calculated as a monthly average.
5. *Comments*—a sum of the three distinct comment events: `IssueComment Event`, `CommitCommentEvent`, and `PullRequest ReviewCommentEvent`; for the previous three months, calculated as a monthly average.
6. *Authors*—the total number of unique authors by username that engaged with the repository for the previous three months, calculated as a monthly average of the following event-types: `PullRequestEvent`, `IssuesEvent`, `IssueCommentEvent`, `PullRequestReviewCommentEvent`, and `PushEvent`.
7. *Contributors*—the total number of unique authors that engaged with the repository since its creation of the same event-types as *Authors*.
8. *Days Inactive*—the number of days since the repository received an update, e.g. 0.5 means the repo was updated 12 hours before the query date. This metric is reversed-scored to be positively valenced, i.e. directionally consistent in the manner that a high value is positive. The maximum value in Table 1 of 1314.6 means the repository was updated within five minutes of the query date.

4.3. Statistical Assumptions

The statistical assumptions that are made before conducting the factor analysis involve the distribution of the data and the correlation matrix (Figure 1). For factor analysis, normality is not required and multicollinearity can assist with factor derivation [16]. The data here are highly non-normal, the Cullen and Frey plots showing most variables to have a kurtosis and skew relationship consistent with data in the beta distribution range.

Tests to justify the application of EFA to the correlation matrix include Bartlett's test of sphericity and the Kaiser-Meyer-Olkin (KMO) test [29], also called the measure of sampling adequacy. Bartlett's test provides a single value for correlation of the data within the matrix, the result of $\chi^2(393) = 4400,\ p < 0.000$ indicates the data is suitable for EFA. Bartlett's test is sensitive to sample size, and so the KMO test is also used. The KMO test is a score between 0 and 1 with 1 representing

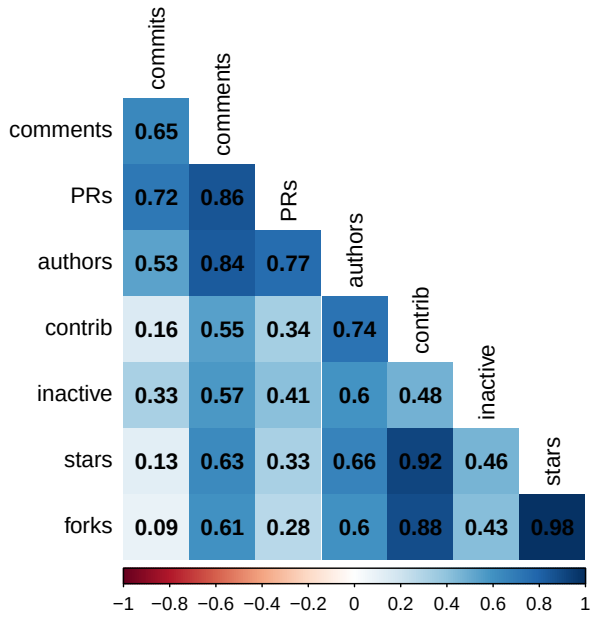

Figure 1. The Pearson correlation matrix for the dataset shows many strong correlations as input data for exploratory factor analysis.

perfect correlation among the variables. Here the overall KMO is 0.77 and each indicator variable is above the 0.5 threshold and thus the data is not random and a candidate for EFA.

5. Results

Prior to running factor analysis on the dataset, scree plots are investigated. EFA is run many times with varying estimator and rotation methods and with a single- and two-factors to derive the number of factors (Section 5.1) and iterate. Interpretation of the results is found in Section 5.2, and validation follows in Section 5.3.

5.1. Deriving Factors

An a priori criterion of a single factor representing developer engagement is the basis of the factor derivation. Secondary factors are not excluded and investigated as part of the EFA. Multiple iterations of factor analysis were undertaken as required, such as for the removal of indicator variables.

Scree plots have been examined to see the number of proposed factors, keeping in mind that they have been shown to be inaccurate for determining the number of factors to retain [15]. A better method is to use a parallel analysis (PA) which employs random data with the same properties (mean, std, etc.) to compare with. Should there be strong deviation of the actual from the simulated data, then a statement can be made in support for the number of factors. Both a standard scree and parallel analysis (Figure 2) indicate strong preference for a single factor and weak preference for a second factor. The PA has significant deviation from the random data at eigenvalue $\lambda_1 = 4.63$ for the first factor and $\lambda_2 = 1.23$ for the second compared to simulated values of 0.48 and 0.16 respectively.

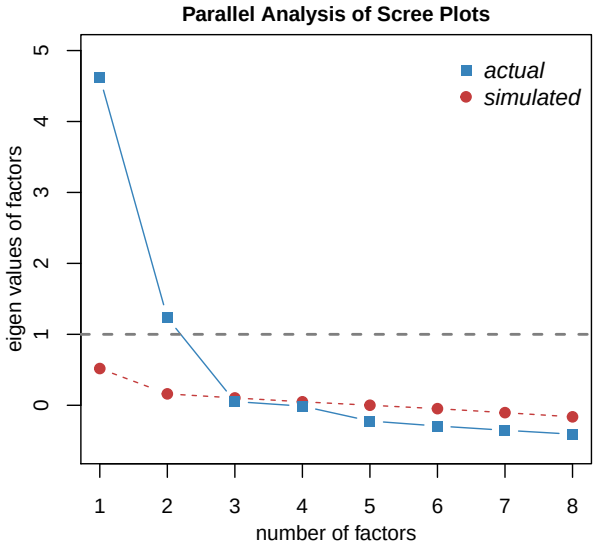

Figure 2. Parallel analysis results indicating strong support for a single factor ($\lambda_1 = 4.63$) and mild support for a second factor ($\lambda_2 = 1.23$).

Factor analysis was carried out with the *Psych* package (version 2.2.3) in R (version 4.0.2). The computational method used to estimate the factors is maximum-likelihood, or ML, known to perform well when the factor-variable relationships are strong. The principle-axis method was also used for comparison purposes as it is ideal for non-normality and small sample sizes [17]. The models for both methods were similar; only results for ML are reported here.

Factor rotation is done with the *GPArotation* package (version 2022.4-1).The VARIMAX factor rotation method maximizes the variances of the loadings within the factors. This can help with structure for two or more factors. Conversely the QUARTIMAX method is a better choice for detecting an overall factor as it maximizes the squared loadings so that each item loads most strongly onto a single factor. For this reason, QUARTIMAX rotation is used with resulting stronger loadings applied to the first factor. Both QUARTIMAX and VARIMAX are orthogonal rotation methods and therefore the amount of variance explained by each factor is reported.

Model A. Table 2 shows the loadings for a two-factor ML with QUARTIMAX rotation. Loading signifi-

Table 2. Model A: Two-factor analysis showing the significant loadings in boldface. Stars and forks exhibit outsized influence on the model.

Factor	A1	A2	h^2
stars	***0.99**	0.16	0.995
forks	***0.98**	0.12	0.968
contrib. total	**0.90**	0.18	0.841
days inactive	0.40	0.41	0.333
pull requests	0.18	**0.94**	0.920
comments	0.51	**0.81**	0.915
commits	0.00	**0.77**	0.599
authors	0.56	**0.70**	0.804
SS loadings	3.50	2.88	
cumulative var.	0.44	0.80	
proportion expl.	0.55	0.45	

Table 3. Model B: Two-factor analysis after stars and forks have been removed. Significant loads are in boldface.

Factor	B1	B2	h^2
pull requests	**0.96**	-0.03	0.078
comments	**0.90**	0.22	0.140
authors	**0.81**	0.47	0.121
commits	**0.74**	-0.14	0.428
contrib. total	0.38	**0.92**	0.014
days inactive	-0.13	-0.02	0.984
SS loadings	3.10	1.13	
cumulative var.	0.52	0.71	
proportion expl.	0.73	0.27	

cance thresholds of > 0.5 are in boldface. The asterisks show strong influence of stars and forks.

The high communality, h^2, (and therefore low uniqueness) of the variance attributed to stars and forks suggests these indicators could measure the same thing and therefore be removed, or be indicative of a new factor. This is the motivation for the next round of factor analysis.

Model B. Table 3 shows the factor loadings and communalities after removing the indicators stars and forks. Using the same estimator and rotation method, the significant loadings suggest a primary factor composed of pull requests, comments, authors, and commits.

Individual fit statistics are not hard rules, nor provide enough evidence to be used in isolation and therefore it is recommended to use a suite of tests for comparison [15]. Fit statistics for the models are summarized in Table 4.

As this EFA is part of a discovery process for CFA/SEM, the CFA thresholds can be applied. Chi-squared: Model A is $\chi^2 = 493.4, p < 0.000$ which is insignificant and suggests the model could be better. This is a badness-of-fit indicator and $p > 0.05$ is significant as in Model B ($\chi^2 = 7.289, p < 0.12$).

Table 4. Fit statistics for the two models showing better overall fit for Model B

Fit Statistic	Model A	Model B
χ^2	493.4, $p < 0.000$	7.289, $p < 0.12$
TLI	0.763	0.993
RMSEA	0.307	0.046
SRMR	0.03	0.01
BIC	415.74	-16.61

The Tucker-Lewis Index (TLI) compares the resultant model with a baseline where there is no factor structure. Model B is significantly better and meets the > 0.9 target [15], and > 0.95 target for large sample sizes ($n > 250$) [16].

The root mean square error of approximation (RMSEA) < 0.05 is a common level however this should not be a universal cutoff [30]. Hair revises upwards to < 0.07 [16] as applied more to structural equation models, further recommending that model fit is better represented by a TLI or comparative fit index measure.

SRMR is the square root of the sum of squared correlation residuals for the variables. Also a badness-of-fit measure, SRMR> 0.1, indicates a good model fit, however, this is biased upward so can be ignored with few indicators as in this study [15].

The Bayesian information criterion (BIC) is a comparative indicator where lower values are better. Model B's BIC is less than model A, supporting the other fit statistics, and providing evidence that the exclusion of stars and forks produces a better model.

5.2. Interpretation

The first model (Table 2) suggests a primary factor (A1) composed of: *stars, forks*, and *total contributors* with a secondary factor (A2) being: *PRs, comments, commits,* and *authors*. Levels of significance are determined with loadings > 0.5; all loadings are shown for completeness.

The high communality, h^2, (and therefore low uniqueness) of the variance attributed to stars and forks suggests these indicators could measure the same thing and therefore be removed, or be indicative of a new factor. The sum or squared loadings for each factor (SS loadings) indicates stronger relationship for A1 (3.50) compared to A2 (2.88).

The model is revised to exclude *stars* and *forks* with the intent of reserving these indicators for an additional

factor. Table 3 shows the revised factor analysis that retains the indicator variables from A2 (Table 2), as the primary factor (B1) with a much higher SS loading than the secondary factor (B1=3.10 compared with B2=1.13). There is moderate cross loading on *authors*, as it has near-significant value of 0.47 loading on B2. As the loading on B1 is more significant, and B2 is of secondary interest in this study, this is acceptable. The cumulative variance explained by the two-factor model is 71% with B1 responsible for 73% of that.

Model A showed a reasonably significant cross-loading and equal influence of *days inactive* on the factors (≈ 0.4). This indicator was left in for Model B and the revised influence is nil.

5.3. Validation

Model validation is by two mechanisms: confirmatory factor analysis applied to the measurement model, and cross-validation by separation of the dataset into a training and testing segment. CFA is undertaken as part of a larger study on software health, and thus cross-validation is most relevant to the present work.

Cross validation is necessary to avoid the situation where the model ends up being overfit to the data, affecting generalizability. Constraints in the data collection process on the number of available projects limit collecting an entire new dataset and so the original is split into two groups. Random allocation is performed using the *caTools* package (version 1.18.0) with 50% split to produce two groups–one to build the model and one to test the model. This split allows for half the data to be close to the minimum sample size threshold of 200.

Table 5 shows the validation results. The insignificant loadings are not shown for readability and to highlight that the same factor structure is present across the models. The training and testing models are not as well fit as Model B based on χ^2, TLI, and RMSEA, however are acceptably close considering the sample size handicap.

6. Discussion

The highest loaded indicators often directly endow their etymology to the factor. Here, the researcher must be able to adequately name the factors, which is subject to interpretation and criticism [16].

Factor A1 fits naturally as a community indicator of *Popularity*, or *Community Interest*, and is distinguished in the model from the more traditional developer activities. *Developer Engagement* remains the best naming option to capture A2's latent construct as these events are more likely to be part of a developer's contribution than casual a community member's.

It is for this reason a two-factor analysis is used to be able to judge where to draw the line for our a priori condition of developer engagement. Does forking count as engagement or could it be better represented by something else? Both models A and B conclude the same indicator variables for the dimension with second factor components being responsible for the statistical fit differences.

The high correlation of stars and forks has been known in the broader software environment when assessing GitHub's overall most popular projects [9, 13]. Figure 1 shows the correlation at 0.98 and this confirms the stars/forks relationship for blockchain projects. Starring and forking are not active developer activities, rather are passive; many people browsing by a project may star it to bookmark the project, or fork it to explore the code. Both activities are easy to do–starring requires a single click and forking needs two clicks. They are much less intensive than commenting on an issue or submitting a pull request and can thus be categorized as a general population GitHub activity, whereas authoring updates is a classic developer activity. Thus stars and forks are removed from the analysis leading to Model B diagrammed in Figure 3. The factor analysis was also done with the inclusion of forks (more or a developer-based activity) and exclusion of stars (more social-media-based activity), and vice-versa. This did not affect the factor structure; stars and forks stood on their own as an independent factor.

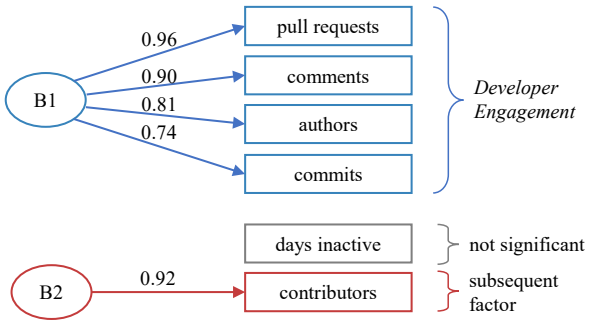

Figure 3. Factor analysis diagram; B2 leads to a subsequent factor and B1 is named Developer Engagement.

Both Model A and B indicate total contributors should be a second factor and this will be taken under consideration in future work. Model A suggests this dimension to include stars and forks. A potential candidate name is *Public Interest* whereby if a project becomes interesting or popular, developers will bookmark it (star or fork) and author a comment (which is counted in total contributors). However, the strength of

Table 5. Cross-validation for Model B showing equivalent factor structure for the testing model as compared to the base EFA. Insignificant loadings are not shown.

	Model B		Training		Testing	
Factor	B1	B2	Tr1	Tr2	Te1	Te2
commits	0.74		0.91		0.70	
comments	0.90		0.94		0.99	
pull requests	0.96		0.97		0.96	
authors	0.81		0.84		0.89	
contrib.total		0.92		0.41		0.87
days inactive						
n	393		195		196	
χ^2	7.289		9.77		25.74	
TLI	0.99		0.98		0.92	
RMSEA	0.046		0.086		0.166	

the stars-forks bond may exclude contributor numbers all together, leaving an under-determined dimension.

The number of days since a repository last had an update (days inactive) does not influence either factor. In both models this indicator did not have a preference for either factor, and in Model B its loadings were -0.13 and -0.02, both insignificant. As this is a time-base metric it may apply more accurately to time-critical dimensions such as time to resolve issues or merge pull-requests.

The strongest loading factor in Model A, Model B, and the cross-validation training model is pull-requests. Although not singularly dominant, the PR (in addition to issues tracking—comments) is seen as the primary way to update a codebase [2]. Given the ease with which GitHub has facilitated the pull-based model [31], it is justified to conclude that developer engagement is led by pull-requests and commenting metrics.

To summarize, the statement of influence from Section 3.1 can now be written as: developer engagement is strongly positively related to pull requests and comment activity. Authors and commits also present positive influences, all on a recent three-month timeline.

7. Threats to Validity

It must be noted that "factor analysis will always produce factors" and is agnostic to "garbage in, garbage out" [16]. Therefore, the role of the researcher is to ensure appropriate indicators are chosen, data is carefully collected, and factor results are thoughtfully interpreted.

Beginning with data collection, a few threats are notable. First, the data is only sourced from GitHub. This is due to the prominence of blockchain projects being hosted here, but must be considered. Secondly, repository owners can move and rename repos in the time between manual verification of the location and the database query time. In this instance they may be missed or counted twice.

Regarding the stargazer event, this is a unidirectional metric as there is no event for removing a star. Thus the reported number of stars by querying the database may be slightly larger than what appears on the website. In practice, few people remove a star. Large sample sizes will mitigate these issues as the model is interested in capturing trends across many projects.

On fit statistics used to verify the models (e.g., RMSEA, and TLI), the thresholds commonly recommended were developed in the context of normally distributed data [15], and must not be considered law. Acceptance or rejection of models should not be based on fit statistics, rather on the ability of the model to provide structure to the data.

8. Conclusion

This work uses exploratory factor analysis to identify dimensions that represent engagement in a community of open source blockchain developers. Findings indicate that *developer engagement* consists of *authors*, *commits*, *comments*, and *pull requests* based on a monthly average of the previous three months. Pull requests is the single strongest influence indicator. *Stars*, *forks*, and *total contributors* are strongly loaded on a second factor hypothesized to be *Public Interest*. Cross validation of the dataset is carried out showing the same factor structure for developer engagement with similar fit characteristics.

The next step is to build a structural equation model of software health that incorporates developer engagement among other indicators such as public interest and project robustness.

References

[1] GitHub, "The 2021 State of the Octoverse." https://octoverse.github.com/, 2021.

[2] G. Gousios, M. A. Storey, and A. Bacchelli, "Work practices and challenges in pull-based development: The contributor's perspective," *Proceedings - International Conference on Software Engineering*, vol. 14-22-May-, pp. 285–296, 2016.

[3] Y. Hu, J. Zhang, X. Bai, S. Yu, and Z. Yang, "Influence analysis of Github repositories," *SpringerPlus*, vol. 5, no. 1, 2016.

[4] P. Poba-Nzaou and S. Uwizeyemungu, "Worries of open source projects' contributors: Patterns, structures and engagement implications," *Computers in Human Behavior*, vol. 96, no. September 2018, pp. 174–185, 2019.

[5] Y. Fang and D. Neufeld, "Understanding sustained participation in open source software projects," *Journal of Management Information Systems*, vol. 25, no. 4, pp. 9–50, 2008.

[6] M. Shaikh and N. Levina, "Selecting an open innovation community as an alliance partner: Looking for healthy communities and ecosystems," *Research Policy*, vol. 48, no. 8, p. 103766, 2019.

[7] A. Bosu, A. Iqbal, R. Shahriyar, and P. Chakroborty, "Understanding the motivations, challenges and needs of blockchain software developers: A survey," *Empirical Software Engineering*, vol. 24, pp. 2636–2673, 2019.

[8] B. Perens, "The Open Source Definition," in *Open Sources: Voices from the Open Source Revolution* (C. DiBona, S. Ockman, and M. Stone, eds.), pp. 171–188, O'Reilly & Associates, 1999.

[9] M. A. Abdulhassan Alshomali, *Open source software GitHub ecosystem: a SEM approach*. PhD thesis, James Cook University, 2018.

[10] U. Raja and M. J. Tretter, "Defining and evaluating a measure of Open Source Project survivability," *IEEE Transactions on Software Engineering*, vol. 38, no. 1, pp. 163–174, 2012.

[11] Z. Wang, Y. Feng, Y. Wang, J. A. Jones, and D. Redmiles, "Unveiling Elite Developers Activities in Open Source Projects," *ACM Transactions on Software Engineering and Methodology*, vol. 29, no. 3, 2020.

[12] D. A. Tamburri, F. Palomba, A. Serebrenik, and A. Zaidman, *Discovering community patterns in open-source: a systematic approach and its evaluation*, vol. 24. Empirical Software Engineering, 2019.

[13] H. Borges, A. Hora, and M. T. Valente, "Understanding the factors that impact the popularity of GitHub repositories," *Proceedings - 2016 IEEE International Conference on Software Maintenance and Evolution, ICSME 2016*, no. Dcc, pp. 334–344, 2017.

[14] M. Clark, "Structural Equation Modeling." https://m-clark.github.io/sem/, 2018.

[15] W. H. Finch, "Using Fit Statistic Differences to Determine the Optimal Number of Factors to Retain in an Exploratory Factor Analysis," *Educational and Psychological Measurement*, vol. 80, no. 2, pp. 217–241, 2020.

[16] J. F. Hair Jr., W. C. Black, B. J. Babin, and R. E. Anderson, *Multivariate Data Analysis*. Essex: Pearson Education Limited, seventh ed., 2014.

[17] M. W. Watkins, "Exploratory Factor Analysis: A Guide to Best Practice," *Journal of Black Psychology*, vol. 44, no. 3, pp. 219–246, 2018.

[18] L. Fabrigar and D. Wegener, *Exploratory Factor Analysis*. Oxford University Press, 2012.

[19] M. Goeminne and T. Mens, "Analyzing ecosystems for open source software developer communities," in *Software Ecosystems: Analyzing and Managing Business Networks in the Software Industry* (S. Jansen, ed.), no. 2013, ch. 12, pp. 247–275, 2013.

[20] A. Milovidov, "Everything You Ever Wanted To Know About GitHub (But Were Afraid To Ask)." https://ghe.clickhouse.tech/, 2020.

[21] H. Aguinis, R. K. Gottfredson, and H. Joo, "Best-practice recommendations for defining, identifying, and handling outliers," *Organizational Research Methods*, vol. 16, no. 2, pp. 270–301, 2013.

[22] L. S. Fidell and B. G. Tabachnick, *Preparatory Data Analysis*, ch. 5, pp. 115–141. John Wiley & Sons, Ltd, 2003.

[23] J. Schroer and G. Hertel, "Voluntary engagement in an open web-based encyclopedia: Wikipedians and why they do it," *Media Psychology*, vol. 12, no. 1, pp. 96–120, 2009.

[24] J. Coelho, M. T. Valente, L. L. Silva, and A. Hora, "Why we engage in FLOSS: Answers from core developers," *Proceedings - International Conference on Software Engineering*, pp. 114–121, 2018.

[25] A. Lee, J. C. Carver, and A. Bosu, "Understanding the Impressions, Motivations, and Barriers of One Time Code Contributors to FLOSS Projects: A Survey," *Proceedings - 2017 IEEE/ACM 39th International Conference on Software Engineering, ICSE 2017*, pp. 187–197, 2017.

[26] G. Pinto, I. Steinmacher, and M. A. Gerosa, "More common than you think: An in-depth study of casual contributors," *2016 IEEE 23rd International Conference on Software Analysis, Evolution, and Reengineering, SANER 2016*, vol. 1, no. 1, pp. 112–123, 2016.

[27] H. Hata, N. Novielli, S. Baltes, R. G. Kula, and C. Treude, "GitHub Discussions: An exploratory study of early adoption," *Empirical Software Engineering*, vol. 27, no. 1, pp. 1–32, 2022.

[28] E. Kalliamvakou, G. Gousios, K. Blincoe, L. Singer, D. M. German, and D. Damian, "An in-depth study of the promises and perils of mining GitHub," *Empirical Software Engineering*, vol. 21, no. 5, pp. 2035–2071, 2016.

[29] H. F. Kaiser, "An index of factorial simplicity.," *Psychometrika*, vol. 39, pp. 31–36, 1974.

[30] F. Chen, P. J. Curran, K. A. Bollen, J. Kirby, and P. Paxton, "Inequality convergence: How sensitive are results to the choice of data?," *Sociological Methods & Research*, vol. 36, no. 4, pp. 462–494, 2008.

[31] G. Gousios, M. Pinzger, and A. V. Deursen, "An exploratory study of the pull-based software development model," *Proceedings - International Conference on Software Engineering*, no. 1, pp. 345–355, 2014.