Health is Wealth: Evaluating the Health of the Bitcoin Ecosystem in GitHub

Khadija Osman
School of Computer Science
Carleton University
Ottawa, Canada
khadija.osman@carleton.ca

Olga Baysal
School of Computer Science
Carleton University
Ottawa, Canada
olga.baysal@carleton.ca

***Abstract*—Bitcoin is a virtual and decentralized cryptocurrency that operates in a peer-to-peer network providing a private payment mechanism. It is a multi-billion dollar cryptocurrency, and hundreds of other cryptocurrencies are created based on it. Bitcoin is based on open source software (OSS) development. This paper presents the first comprehensive study of the Bitcoin ecosystem in GitHub organized around 481 most popular and actively developed Bitcoin related projects over eight years (2010–2018). Our work includes manual categorization of the projects, defining software health metrics, classification of projects according to these health metrics, and evaluation of the health trends of the ecosystem. The main findings suggest that the Bitcoin ecosystem in GitHub is represented by nine categories of projects. Moreover, the health of the majority of the projects is assessed as "Low Risk".**

***Index Terms*—Bitcoin, cryptocurrency, ecosystem, health metrics, code quality, classification, trends**

I. Introduction

The invention of Bitcoin as a currency has introduced an information technology that facilitates secure and decentralized payment systems and tools for storage, verification, and auditing the information. Bitcoin is a decentralized entity that is regulated by its users. Furthermore, it provides online transfers between two individuals based on cryptographic proofs rather than trust. Bitcoin relies on cryptographic algorithms and a distributed network of users, allowing them to mine, store and exchange their Bitcoins. Since the invention of Bitcoin in 2008, it has become the fastest growing and most trusted cryptocurrency in the market [1].

The concept of software ecosystem was first introduced in 2003 [2]. Since then, many different definitions of software ecosystems were proposed [2], [3]. In this work, we adopt the definition of a software ecosystem as proposed by Jansen et al. [4]: "software ecosystem a set of projects, and actors functioning around a common technological platform or a market; they exchange knowledge, resource, and artifacts". Thus, our definition is also concerned about projects and people involved in developing them for the purpose of sharing their knowledge, expertise, and contributions. Our goal is to define the Bitcoin software ecosystem in GitHub in terms of its available projects and its contributors. Furthermore, we perform an analysis of the health of the defined Bitcoin ecosystem.

Currently, the total supply of Bitcoins available is 21 million, and the reward for Bitcoin mining is 6.25 Bitcoins. However, the reward for Bitcoin mining is cut in half every four years leading to a regular decrease in the incentive to mine Bitcoin, and with it a decrease in the incentive to develop software around Bitcoin as the reward to do so diminishes. Furthermore, another concern that miners face is related to how they would be paid if the supply of Bitcoin overruns. One possible solution would be charging a transaction fee. Nevertheless, Bitcoin remains the leading cryptocurrency in the market, so it is important to explore its software ecosystem and investigate whether it has formed a healthy software ecosystem. The motivation behind studying this ecosystem in GitHub is that GitHub is the most popular collaborative development platform. GitHub is hosting more than 10 million software repositories, and millions of developers collaborate on project development in GitHub. Moreover, Bitcoin's main project is hosted on GitHub. Our study can provide useful insights to the developers and researchers in the field of OSS development, cryptocurrency, and software ecosystem.

The objective of this study is two-fold: 1) we are interested in defining the structure of the Bitcoin ecosystem in GitHub by categorizing the Bitcoin related projects, and 2) we aim to evaluate the health of the defined Bitcoin ecosystem and to analyze the trends in the health of the ecosystem as a whole.

In this study, we address the following research questions:

- **RQ1: How the Bitcoin ecosystem is represented in GitHub?** Bitcoin is the leading cryptocurrency, understating its development in GitHub can help us define its open-source software ecosystem (OSSE).
- **RQ2: How healthy is the Bitcoin ecosystem in GitHub?** Software is central to the Bitcoin function, and it follows OSS development. Maintaining a healthy ecosystem of OSS is also a prerequisite for the sustainable development of open-source communities.
- **RQ3: What are the trends in the health of the Bitcoin ecosystem?**
  Answering this question can provide us with the indication of whether the health of the ecosystem is improving overtime or deteriorating.

This paper makes the following contributions:

1) Mapping out the Bitcoin ecosystem in GitHub by collecting and analyzing all the relevant projects.
2) Analysis of the trends in the ecosystem's health based on the four metrics such as popularity, maturity, activity, and code quality.
3) A set of recommendations to researchers and developers in the field of cryptocurrency and software ecosystem.

II. Related Work

A. Software Ecosystem

There is an extensive research in the field of software ecosystems (SEs) in the past 10 years. Manikas et al. [5] found that there is a significant increase in the published papers in this field each year form 2007 to 2014. Some researchers consider such rapid evolution of the large ecosystems as one of the reasons, beside the product-line, and global development, for a growing complexity in software development [6]. The research state-of-the-art in this field is organized around three main concepts such as defining the architecture of SEs, identifying new SEs, evolution and health of SEs.

1) Software ecosystem's architecture/identifying new SEs: There is a vast number of studies towards defining the architecture of a software ecosystem. Hyrynsalmi et al. [7] have summarized the definitions in the literature into three groups. The first group sees the software ecosystems (SE) as a special case of business ecosystem, similar to Jansen et al.'s view [4]. Many researchers agree to this definition that SEs should be business oriented [4], [8], [9]. The second group considers SEs as open-source projects, its dependencies and community [10]. The last group defines SE as a set of developers or organizations working together on a set of development projects. Some researchers have conducted user studies with experts in this domain to develop a definition of a software ecosystem [5], [11]–[13]. Our research focuses on exploring how a well-known Bitcoin ecosystem is represented and defined in GitHub. Following the second group of SE, we define the structure of the Bitcoin ecosystem in GitHub in terms of the Bitcoin repositories, the developer community, and a shared framework.

2) Software ecosystem's health: The health of a SE defined by Jansen et al. [14] refers to the normal functioning of the projects that constitute the ecosystem and of the ecosystem as a whole. They identified productivity, robustness, niche creation as the health indicators of an SE. Furthermore, a group of researchers [15] adopted a socio-technical perspective. Following this perspective, they considered three levels of health: social, technical and phenomenological. They concluded that the health problems propagate to the ecosystem due to the social and technical dependencies between the projects [16]. They identified three social problems such as loss of contributors, lack of interactions between contributors, and low number of contributors and two technical problems: poor code quality, and low number of code commits. Besides, some researchers used the metrics [14] to evaluate the health of successful organizations [17], [18]. Similarly, some researchers have investigated the challenges related to the quality assurance of software ecosystems which is critical for embedded systems [19]. There is a number of studies investigating software ecosystem health [7]–[9], [20]–[22].

3) Software ecosystems in GitHub: Majority of the studies around identifying and studying the evolution of a software ecosystem is in the domain of open-source development. There are many studies that identify different ecosystems in GitHub [23]–[27]. To the best of our knowledge, we are aware of no work that studies the Bitcoin software ecosystem and its health in GitHub or other collaborative platform.

III. Methodology

A. Data mining

A subset of the dataset used for this research comes from GHTorrent. We retrieved a number of metrics such as the number of commits, issues, watchers, forks, contributors and programming languages from the GHTorrent dataset. However, GHTorrent does not store the data related to source code of the repositories. To collect source code features we used PyDriller [28] to clone each of the selected repositories and collect source code metrics. We use "repository" and "project" interchangeably in this paper.

1) Project selection and pre-processing: We used GHTorrent's MySQL database to identify Bitcoin related projects. We queried the database via Google BigQuery using the keyword "Bitcoin" in the name and description fields of a project. This query resulted to 80,453 projects. Further, we cleaned the dataset by applying the following filters:

1) We removed all "deleted" projects, as a result, the dataset was reduce to 74,628 projects.
2) We considered only "master" repositories. This filter reduced our dataset to a significant number, i.e., 16,833 projects. This shows that the majority of the Bitcoin projects are forks of the existing projects.
3) Furthermore, we considered projects with at least 10 commits. We ended up with 3,601 projects having more than 10 commits.
4) We focused on "active" projects. Thus, we selected the projects that have their last commit made in the last six months. This filter limits our dataset to only active or ongoing projects.
5) Our last filter eliminated solo projects, thus we consider projects that are maintained by more than two developers. This filter reduced the dataset to 581 projects.

2) Collecting source code features: In order to analyze the quality of the source code, we need to extract features such as code complexity, methods count, token count, etc. To collect these code quality features we used PyDriller [28] to clone each repository in the dataset and extract the required features. Out of 581 repositories 50 of them were not not found during the cloning process. We confirmed this by visiting the GitHub page for each repository, and found that these projects no longer exist. Furthermore, during our semi-automated classification of the repositories we found 50 repositories to be irrelevant, they are not Bitcoin repositories even though the

Table I
FEATURES OF THE REPOSITORY-LEVEL DATASET.

Feature	Description
project_id	Unique ID for each repo
name	Name of the repo
URL	GitHub API URL of the repo
description	Description for the repo
language	Top 3 most frequent prog. languages
created_at	Date the repo is created at
forked from	ID for its parent repo
deleted	Whether repo is deleted or not
updated_at	Date the repo is last updated
forked_commit_id	ID of the commit when repo was forked
commit	Number of commits per repo
issues	Number of issues per repo
watchers	Number of watchers per repo
contributors	Number of contributors per repo
forks	Number of forks per repo
cyclomatic complexity	Mean of CC over the commits in repo
nloc	Mean of nloc in *commits*/**repo**
lines_added	Mean of lines added to *commits*/**repo**
lines_removed	Mean of lines removed from commits/repo
token_count	Mean of token count over the *commits*/**repo**
method_count	Mean of method count over *commits*/**repo**
modifications	Number of modification per *commit*/**repo**
frq_change_type	Frequently changed type in *commits*/**repo**
frq_file_changed	Frequently changed file in *commits*/**repo**

name of the repository contains a "Bitcoin" word. Therefore, we have collected the source code features for 481 repositories over the period of 8 years. As result, our dataset consists of 337,008 commits from 481 GitHub projects. Thus, we ended up having two final datasets: repo-level and commit-level. The repo-level dataset is comprised of 481 projects with 24 features over the 8 years shown in Table I. While the commit-level dataset contains 12 features from cyclomatic complexity to frq_file_changed in Table I over the period of 8 years.

B. Defining the Bitcoin ecosystem

We are interested in determining which Bitcoin applications have open-source projects on GitHub. For this purpose, we manually classify the projects in the dataset based on their application category. Manual classification of the projects was performed by two coders; while disagreement cases were rare, such cases were discussed and resolved by the coders to reach a consensus. We studied the projects' name, description and README file. Initially, we started with *development*, *wallet* and *payment* categories. These are the prominent categories in the more general Bitcoin ecosystem [29], so we anticipated their presence in GitHub. We searched the GHTorrent dataset for Bitcoin projects which could be a part of a certain category. For example, we knew that Bitcoin has its official repository hosted on GitHub, and is a part of the *development* category. Similarly, for *payment* category, there are many businesses that are accepting Bitcoin as payment, thus many applications of this nature are likely to be hosted and maintained in GitHub. Furthermore, after searching for Bitcoin wallets we were able to find that some very popular wallet applications such as Bitcoin Core wallet, Samourai, and Armory have GitHub projects. Thus, we first identify all the projects in the dataset that would fall into these three categories. This semi-automated process follows two steps: 1) regular expressions

Table II
KEYWORDS USED.

Category	Keywords
Development	implementation,ide,protocol,plugin,module,algorithm, api,libraries,tool,blockchain
Wallet	wallet,bitkey,xapo,opendime,wasabi,coldcard,greenaddress, sentinel,electrum,Bitcoin core wallet,samourai,armory
Services	website,Bitcoin cash,proxy,optech,bitnode,otx, Bitcoin visuals,blockstream,coindance,byll,lolli,fold,bazaar, tallycoin,bitpatron,bitrefill,fastBitcoin,purse
Documentation	documentation,doc
Mining	mining,miner,mine,honeyminer,pool,bitfury,canaan, bitmain,ebang,halong mining,sluch pool
Trade	trade,exchange,bakkt,hodl hodl,localBitcoins,cash app, xbt provider,paxful,cme group, bull Bitcoin,Bitcoin international trust
Payment	payment,bitpay,opennode,blockstream satellite, paynym,azteco,btcpay,gotenna
Node	node,casa,nodl,dojo,raspiblitz,lightning,bitseed
Data Analysis	data,analysis,social,media,analytic

were used to search for keywords (shown in Table II) in the name or description of a project, and 2) manual check of the project's GitHub URL to confirm the results.

During the initial search, we found that some of the repositories can be better presented by the categories other than the initial three categories that we stared with. Thus, we have identified six more categories of the ecosystem. Our classification approach includes manual analysis of the repositories with no description or if the type of the repository was not clear from its description. For the repositories with no description (385 projects), we manually classified them using their README files.

C. Health Analysis

We define health metrics and then classify the projects into three classes of health. Furthermore, we analyze the health trends at the project level and the ecosystem as a whole.

1) Health metrics definition and heuristics : In order to classify the projects we introduce four metrics: 1) popularity, 2) activity, 3) code quality, and 4) age and assigned the features that are associated with them. The combination of these metrics serves as proxy of our definition of the ecosystem's health. Table III presents these metrics, labels, related features, and the algorithms used for assigning labels to the projects.

Activity: Projects with a high number of code commits are well developed [15]. We also associate the number of issues created within a repository to the activity of the project, assuming that a high number of issues contributes to higher participation of the greater user community.

Age: Age is an indicator used for determining the health of an open-source project. Projects may be *mature*, *growing* or *new*, i.e., created just a few months ago.

Popularity: We use watchers as the popularity metric. We believe that projects with higher popularity are more likely to be actively developed and maintained. This belief is echoed by the existing work, e.g., Jansen et al. [14] state that the number of project's downloads is a strong predictor of its good health. While Borges et al. [30] found that three out of four users consider the number of stars on the project prior to contributing to it.

Table III
HEALTH METRICS.

Category	Labels	Features	Labeling
Popularity	high medium low	watchers forks	Algorithm 1
Complexity	high medium low	code_complexity nloc lines_added lines_removed token_count method_count	Algorithm 3
Activity	high medium low	commits issues contributor	Algorithm 2
Age	new growing mature	age_in_weeks	Algorithm 4

Table IV
FEATURES; Q DENOTES "QUARTILE".

Feature	Q1	Q2	Q3	Mean	Max
Commits	44.5	342	914.3	623.86	10,901
Issues	2.25	70.06	527.73	111.12	13,038
Contributors	2.35	5.19	128.28	6.5	442
Watchers	10.45	122.53	778.41	223.06	32,056
Forks	6.39	47.97	538.94	82.95	1,4951
Age	15.79	48.03	51.56	51.63	380
CC	4.51	30.23	211.54	34.72	1,929.77

Code quality: By considering code complexity metric, we can assess code quality of the project. If the complexity of the source code is high, it may contribute to a poor quality of code. We use cyclomatic complexity (function level) [31] to measure the code complexity due to ease of calculation and interpretation. It supports major languages such as C/C++, Java, C#, JavaScript, Objective-C, Python, and more.

2) Classification of projects: Using quartiles as thresholds is a common practice in different fields [32]. Similarly, for labeling, we considered quantiles as the thresholds instead of the mean, because the mean can be affected easily by the outliers in the data. Table IV shows the descriptive statistics of the features in the ecosystem. Algorithms 1–5 are heuristic-based approaches of assigning labels (e.g., *high*, *medium*, or *low* popularity) to the projects.

Algorithm 1: a heuristics-based approach of classifying projects based on the popularity metric. If the value of watchers is less than the (Q1) then its popularity is determined as *low*. While if it is greater than (Q1) and less than (Q2) we label the project's popularity as *medium*. While if it is greater than (Q2) then we label the project's popularity as *high*.

Algorithm 2: a heuristic-based classification based on the project activity. If the values of all the features are greater than Q1, then we label the project's activity as *high*. Furthermore, if two of the features have a value greater than the Q3, we label the project as having *high* activity. On the other hand, if any of the features have a value less than Q1, then we label the project as *low*. For the class *medium* activity; two of the features should have a value greater than Q1.

Algorithm 3: a heuristic-based classification based on the age of the projects. We set three thresholds. If the value of age is less than Q1 then we label the project as *new*. While if the value of age is greater than Q1 and less then Q2, we label the project as *growing*. On the other hand, if the value of age is greater than Q2 then we label the project as *mature*.

Similarly, *Algorithm 4* is a heuristic-based approach of classifying projects based on their code complexity. Here again, we set three thresholds. First, if the value of the complexity is less than Q1, then we label the project as *low* in terms of complexity, but if the value is between Q1 and Q2 the project is labeled as having *medium* complexity. Lastly, for all other cases the project is classified as having *high* complexity. While health can be defined as spectrum of different categories, we classify project's health into *healthy*, *low risk*, and *at risk* categories.

Algorithm 5 describes the process of classifying projects into three health categories. If a project shows *low* activity, defined as *new*, with *low* popularity, *high* or *medium* complexity, then we classify the project as *At Risk*. For the second class, if a project demonstrates *high* or *medium* activity, is a mature or growing project, with *high* and *medium* popularity, with *low* and *medium* complexity values, we label the project as *Low Risk*. If none of the above conditions are satisfied, then the project is classified as *Healthy*.

3) Trends in health of the ecosystem as a whole: We define the health of a software project in terms of four metrics such as popularity, activity, maturity (age), and code quality. These metrics combine both technical and social aspects related to the health of an open-source project. We perform a quantitative analysis of the trends considering all the defined health metrics for the ecosystem as whole, not just at the project-level.

Activity: To identify the trends of the ecosystem's health over the years we consider the following:

1) For the analysis of the contributions, we consider the number of commits per year, per category, and a combination of both, and the change in contribution per year.
2) A continuous collaboration between different stakeholders is important for an ecosystem to function well [33]. To analyze collaboration between developers, we analysed the number of authors working in one, two, three, four and five or more projects. For retention rate, we calculate the number of new authors subtracted from the loss of authors every year. We calculate the change of contributors for all the projects per category, per year.
3) To determine the loss of contributors in a year, we consider the authors whose last commit was made in that year.
4) To analyze issues, we determine the average number of issues created each year for all projects and the average number of issues created for each year for all categories.

Popularity: To analyze the popularity, we calculate the mean of watchers for each year for all repositories, as well as the average number of watchers per year and per category.

Code quality: To analyze the quality of the source code, we determine how the CC is changing over the years.

IV. RESULTS

We now present the results and answers to our research questions.

Table V
THE CATEGORIES OF THE BITCOIN SOFTWARE ECOSYSTEM.

Category	#repos	#issues	#watchers	#devs	#commits
Data analysis	25	95	103	80	1,539
Development	223	32,114	79,050	1,661	181,875
Documentation	6	17	23	17	186
Mining	11	41	58	36	2,116
Node	26	6,952	10,030	421	28,065
Payment	27	626	1,067	84	8,580
Services	86	5,366	6,153	387	26,023
Trade	31	2,180	5,501	180	14,578
Wallet	46	6,062	5,309	291	34,173

A. RQ1: How the Bitcoin ecosystem is represented in GitHub?

After a thorough research about the applications of Bitcoin, we classify the projects into nine categories. Table V presents the our results of the categories that represent the Bitcoin ecosystem in GitHub. During our manual classification, 50 out of 581 projects (8%) were excluded from the dataset as they were determined to be irrelevant. We now offer a brief explanation of each category:

Development: Any repository which contains specific implementation of a protocol, IDE, plugin, module, library, API, tool, or algorithm is classified as *development*. This category has the highest number of projects and commits — 223 and 181,875, respectively. Bitcoin's main project, called bitcoin/bitcoin is a part of this category, which is the most active project in terms of collaboration, popularity and age in the ecosystem. The top three programming languages in this category are JavaScript, Python and Swift.

Documentation: The projects containing Bitcoin documentation are classified under this category. There are only six projects that contain documentation for Bitcoin with 17 unique developers contributing to this category. One of the projects in this category explains the security threat model of the Bitcoin cryptocurrency.

Services: Any specific implementation of any kind of service is classified under this category. For example, there are different online resources that offer a variety of services related to Bitcoin and its information. Furthermore, many businesses are adopting Bitcoin as payment, they host their websites to provide any needed information related to their business. The most active project in this category is OpenBazaar Desktop, an interface for OpenBazaar Bitcoin node.

Node: A node is a device, usually a computer or any other electronic device, that participates in running a cryptocurrency network. Casa and Eclair are examples of an application that is used to run a node of the Bitcoin network. This category has 26 projects and 421 contributors.

Wallet: This class contains all the projects specific to the implementation of wallets. This category includes 46 projects and 291 contributors. The two most popular projects in this category are Electrum and Bither. The top three languages in this category are Java, Python and Ruby.

Mining: Mining is the process of adding various cryptocurrency transactions and evidence of mining work to the blockchain ledger. This category includes repositories that implement features specific to the mining process. There are many software applications for mining Bitcoins, for example, Bitcoin miner. The main programming languages in this category are Groff and shell.

Table VI
DESCRIPTIVE STATISTICS FOR HEALTH CATEGORIES.

Feature	Healthy			Low Risk			At Risk		
#projects	85			322			74		
	Min	Mean	Max	Min	Mean	Max	Min	Mean	Max
Commits	12	752.2	5,049	10	676.6	10,901	10	246.9	9,580
Devs	2	9.1	91	2	6.9	442	2	2.2	7
Issues	2	202.4	2,432	0	112.4	13,038	0	1	12
Watchers	3	355.9	7,853	0	239	32,056	0	0.9	7
Age	16.3	127.3	343.6	0.9	41.2	380.0	0.4	10	15.7
CC	0	10.6	29.9	0	33	984.5	4.5	69.7	1,929.8

Trade: Trading Bitcoins means selling or buying coins. There are multiple platforms through which we can trade Bitcoin. MetaTrader5 is a popular multi-functional platform offering state-of-the-art trading capabilities, technical analysis tools, and advanced auto trading systems. This category includes 31 projects and 180 contributors, the top two used programming languages are Java and Elixir.

Data analysis: The projects that are related to the analysis and prediction on Bitcoin data. For example, a project in this category implements an application that could predict Bitcoin market prices based on the discussions on Twitter.

Payment: Many big businesses are adopting cryptocurrencies as a method of payment. BTCPay is an example of payment software, which also has a public repository in GitHub. There are 27 projects in this category which offer different solutions to the payment systems. HTML is used by majority of the projects followed by Javascript and CSS.

Answer to RQ1: Based on our findings, the Bitcoin software ecosystem in GitHub is represented by 481 projects as of 01-04-2018. The classification of these projects resulted in nine categories, including *wallets*, *services*, *trade*, *node*, *mining*, *development*, *payments*, *documentation*, and *data analysis*.

B. RQ2: How Healthy is the Bitcoin Ecosystem in GitHub?

In this work, the Bitcoin software ecosystem in GitHub is classified into three classes of health — *Healthy*, *Low Risk*, and *At Risk* with 85, 322 and 74 projects, respectively. Table VI shows the min, max and mean values for the features we used to classify the projects.

The average number of commits in class *Healthy* is 752.188, while it is 676.618 in class *Low Risk*, and 246.918 in class *At Risk*. Similarly, the average number of contributors in class *Healthy* is 9.141, 6.888 in *Low Risk*, and 2.189 in class *At Risk*. Whereas the average number of CC is 10.610 in class *Healthy*, 29.901 in class *Low Risk*, and 69.718 in class *At Risk*. *Healthy* projects appear to be mature, very active projects (on average around 750,000 commits), with large community of contributors, high popularity, high number of issues (issues are representing requests to fix bugs or add new functionality), and low code complexity. *At Risk* projects can be defined as 2–3 month old repositories with a few or zero watchers, issues,

Table VII
NUMBER OF PROJECTS CREATED EACH YEAR.

Year	2010	2011	2012	2013	2014	2015	2016	2017	2018
Projects	5	14	4	12	30	31	23	218	154

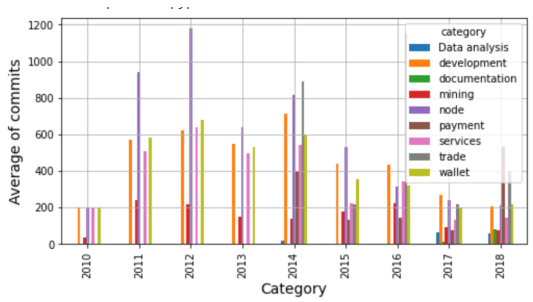

Figure 1. The average number of commits per category and per year.

much smaller community of contributors, and of high code complexity. The *Low Risk* projects are performing well on at least one category of health. For example, this class includes a project which has the maximum number of contributors in the ecosystem, i.e., 442, but the project also has the highest cyclomatic complexity which makes it at *Low Risk* and not *Healthy*.

Answer to RQ2: Majority of the projects of the Bitcoin ecosystem demonstrate *Low Risk* health indicators which is a good sign for the overall health of the ecosystem, while 17% of projects are appear to be *Healthy* and only 15% of projects are deemed to be *At Risk*.

C. RQ3: What Are the Trends in the Health of the Bitcoin Ecosystem?

Here, we present the results of our analysis of the ecosystem's health trends.

1) Activity: Table VII shows the number of projects created each year. Five projects that were created in the year 2010, when Bitcoin has emerged as a cryptocurrency, are still being actively developed. In 2011, 14 more projects have been created, but in 2012 only 4 new Bitcoin related projects were added to GitHub. The number of new projects is increasing after year 2013. In 2014, 30 new projects have been created, and in 2017 the number of new projects has jumped to 218. 2018 is marked with 83 new projects created just in the first 4 months of the year. As we can see from the table the number of projects is increasing rapidly after year 2016. We can observe an increased interest in the Bitcoin related development and representation of projects in GitHub since 2017.

The year-to-year growth rate of commits is decreasing till year 2015 except 2014, where there is a slight increase and the this change is significant from year 2011 to 2012. After 2015 the growth rate is constantly increasing till year 2018. Furthermore, Figure 1 shows the number of commits per each category in the ecosystem. Till year 2014, the Bitcoin ecosystem was represented only via five categories such as *development*, *mining*, *node*, *services*, and *wallet*. In 2010, four categories (except *mining*) have equal number of commits, i.e., 200. Up until year 2013, *mining* category of projects has the lowest number of commits. After 2011, *node* category has the highest number of commits, while there is no significant difference between other four categories. In 2014, *documentation*, *trade*, and *payment* categories have emerged. *Documentation* has the lowest number of commits, while *trade* has the highest number of commits followed by *node*. The *documentation* category shows no activity in 2015 and 2016, but shows slight activity in 2017 and 2018. In 2017, *data analysis* has emerged as a category with the second lowest activity. In 2018, *payment* has the highest number of commits, followed by *trade*, while *development*, *node*, and *wallet* have almost the same number of commits. We also calculated the average number of commits per developer. The average is taken by dividing the total number of commits by the number of projects a developer has contributed to. The number of commits for the majority of the developers is below the average, i.e., 35 commits.

Collaboration of authors: Involvement of different developers is measured in terms of their input to various projects. We calculate the following metrics in order to evaluate the contributions to the Bitcoin ecosystem: the loss of contributors, developers retention rate in the ecosystem, and number of developers contributing across different projects and across different categories.

Additionally, we wanted to explore whether developers within the Bitcoin ecosystem are collaborating and interacting with each other. To determine this we have analyzed developers who are working on more than one project and whether these projects belong to different categories of the ecosystem. We found that 88% of the developers are contributing to only one project, and the remaining 12% are working on more than one project, with 7% of developers contributing to 2 projects, 2% to 3 projects, 1% to 3 projects and 1% of developers to 5 and more projects. The three most collaborating categories are *development* and *services* and *wallet*. where 56 developers are working in both *development* and *services*, and 84 developers are working in *development* and *node*. 84 is the highest number of developers contributing across different categories, following by 76 developers who are working in *development* and *services*. We also found that projects in the *wallet*, *trade* and *development* categories do not have any common contributors.

Issues: The average number of issues created over the years was calculated by dividing the number of issues with the number of projects per year. We found stability in the average number of issues. The number of issues is decreasing constantly after year 2011, meaning that projects' code is becoming less bug prone. Figure 2 shows that *development* is the dominant category in terms of number issues, followed by *wallet* from year 2012 to 2014. After 2015, the most of the issues were created in the *node* category which may appoint to the popularity and higher contributions to this category in the recent years.

2) Popularity: Figure 3 shows the average number of watchers for different categories through the years. In 2010 and 2011, the only category is *development*. After 2012, *wallets* projects are gaining attention, and it becomes the second popular category till 2014. In 2015, *node* is gaining

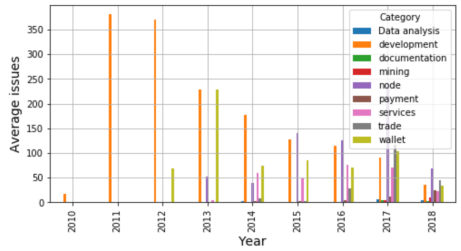

Figure 2. Average number of issues per year per category.

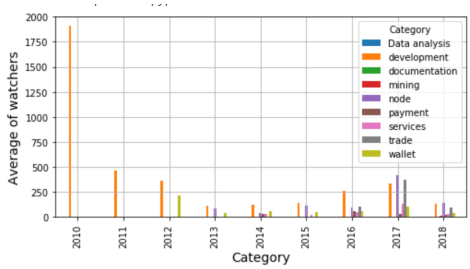

Figure 3. Average number of watchers per year and per category.

more popularity. Between 2015 and 2016, *node* stays as the second popular category, while in 2017 and 2018 it becomes the most popular one followed by *development*.

3) Code quality: The higher the complexity the lower the code quality. High CC means there are more independent paths in a module that would require more tests, as well as higher cognitive loads from developers to comprehend and maintain the system. There is a significant decrease in CC in 2011. From 2011 to 2012 it is increased almost by 100 independent paths through a module. It is decreasing till year 2015, but then it is again increasing till year 2017. Figure 4 shows complexity over the categories. In 2010, all the categories such as *development*, *node*, *services* has the same complexity, while *mining* has the lowest complexity. From year 2011 to 2014, *node* has the highest complexity, while in 2017 it is almost 900 for *development* and around 200 for the rest of the categories. *Mining* overall has the lowest complexity, but in 2018 it reaches the complexity of 1,200 unique paths.

4) Age: Table VIII shows some descriptive statistics about the age (in weeks) of the projects in the ecosystem. The average age of the projects within the Bitcoin ecosystem is 52 weeks (about a year) meaning that most of the projects are relatively young and in early stage of the development. The Bitcoin's main repository is the oldest project in the ecosystem that is 380 weeks (around 7 years and 3 months) old, and is a part of *development* category.

Answer to RQ3: Overall, the Bitcoin ecosystem's activity has been on the rise since 2015; while the developer retention rate has been decreasing. While the ecosystem's popularity and code complexity is growing every year, cross-category contributions and collaboration between developers have been very low.

Table VIII
DESCRIPTIVE STATISTICS FOR PROJECTS' AGE.

Mean	Std	Min	25%	50%	75%	Max
51.634	74.128	0.428	10.428	17.571	44.428	380.000

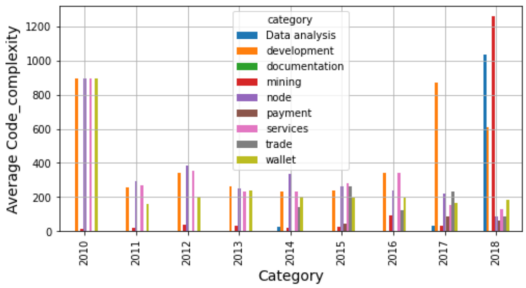

Figure 4. Average code complexity per category.

V. DISCUSSION

A. Implications for the Bitcoin Community in GitHub

We have a number of recommendation to offer for the Bitcoin developers. First, we noticed that the number of contributors seem to be decreasing each year. According to Wahyudin et al. [34], in a healthy and sustainable software ecosystem developers contribute to different projects, but problems arise when they leave the ecosystem. To increase sustainability of the ecosystem, a number of actions can be applied to increase the developer retention rates. Constantinou et al. [35] reveal that developer retention can be improved when contributors remain active and have strong contribution intensity, regardless of whether they are core or peripheral developers. Lin et al. [36] find through survival analysis that developers who join a project earlier have higher chances to stay longer, thus they suggest that encouraging newcomers may effect their survivability on the project. Second, we found only six projects that are related to Bitcoin documentation. To improve code quality, as well as better developer understanding of community's goals, more efforts should be dedicated to writing various kinds of documentation related to code (e.g., code review, code standards), community (existing policies, code of conduct, policies, vision, etc) and more. Third, only 12% of the developers are contributing to more than one category. The Bitcoin ecosystem should facilitate better knowledge transfer across different categories to be able to maintain a sustainable software ecosystem. Having certain initiatives in place, such as a dedicated group of users who can oversee the overall health and growth of the ecosystem can foster more sustainable future of the Bitcoin ecosystem within GitHub.

B. Implications for Research Community

Health metrics of a software ecosystem are not well defined; with most researchers referring to social metrics [15], [16], [37]. Similar to previous work [15], [38], we showed that code quality metrics such as CC, as well as social metrics can be useful in measuring the health of an ecosystem.

We have defined the Bitcoin ecosystem in GitHub using our knowledge of a more general definition of the Bitcoin ecosystem. Many researchers have used other common ways

such as package dependency and project relevance [4], [8]–[10]. However, for ecosystems that may include projects with no known package dependency or relevance such as in the example of the Bitcoin ecosystem, what approach should be followed for identifying such ecosystems and their boundaries? We believe that research community should continue their efforts in establishing and developing techniques for identifying software ecosystems, assessing and evaluating their health.

C. Threats to Validity

One potential threat can be the quality of the GHTorrent dataset. We have mitigated the GHTorrent quality by collecting our metrics by querying "live" GitHub data using Big Query. To mitigate human bias and error during manual categorization for RQ1, two coders were involved in classifying projects into nine categories. Additionally, our categorization of the Bitcoin projects is dated to 01-04-2018; more projects may have emerged in GitHub organized around various and potentially new categories within the ecosystem.

VI. Conclusion

In this work, we have studied the Bitcoin software ecosystem in GitHub and explored its health. We defined the Bitcoin software ecosystem in GitHub by manually classifying the projects into nine categories. We then defined the health metrics by leveraging both the social and technical features of a project. We then analyzed the health trends of the ecosystem by performing the analysis over the span of eight years and across different categories.

References

[1] D. Williams, *Cryptocurrency Compendium: A Reference for Digital Currencies*. Lulu.com, 2017.

[2] D. G. Messerschmitt and C. Szyperski, "Software ecosystems: understanding an indispensable technology and industry. 2003," 2003.

[3] K. Manikas and K. M. Hansen, "Software ecosystems–A systematic literature review," *Journal of Systems and Software*, vol. 86, no. 5, pp. 1294–1306, 2013.

[4] S. Jansen, A. Finkelstein, and S. Brinkkemper, "A sense of community: A research agenda for software ecosystems," in *Int. Conf. on Software Engineering*, 2009, pp. 187–190.

[5] K. Manikas, "Revisiting soft. ecosystems research: A longitudinal literature study," *Journal of Systems and Soft.*, vol. 117, pp. 84–103, 2016.

[6] J. Bosch and P. Bosch-Sijtsema, "From integration to composition: On the impact of sw product lines, global development and ecosystems," *Journal of Systems and Software*, vol. 83, no. 1, pp. 67–76, 2010.

[7] S. Hyrynsalmi, J. Ruohonen, and M. Seppänen, "Healthy until otherwise proven: some proposals for renewing research of software ecosystem health," in *Int. Workshop on Software Health*, 2018, pp. 18–24.

[8] S. Jansen, S. Brinkkemper, and A. Finkelstein, "Business Network Management as a Survival Strategy: A Tale of Two Software Ecosystems," *Iwseco@ Icsr*, vol. 2009, 2009.

[9] J. Bosch, "From software product lines to software ecosystems," in *SPLC*, vol. 9, 2009, pp. 111–119.

[10] M. Lungu, "Towards reverse engineering software ecosystems," in *Int. Conf. on Software Maintenance*, 2008, pp. 428–431.

[11] A. Serebrenik and T. Mens, "Challenges in software ecosystems research," in *European Conf. on Soft. Architecture*, 2015, pp. 1–6.

[12] O. Franco-Bedoya, D. Ameller, D. Costal, and X. Franch, "Open source software ecosystems: A Systematic mapping," *Information and Software Technology*, vol. 91, pp. 160–185, 2017.

[14] S. Jansen, "Measuring the health of open source software ecosystems: Beyond the scope of project health," *Information and Software Technology*, vol. 56, no. 11, pp. 1508–1519, 2014.

[13] J. Knodel and K. Manikas, "Towards a typification of software ecosystems," in *Int. Conf. of Software Business*, 2015, pp. 60–65.

[15] T. Mens, B. Adams, and J. Marsan, "Towards an interdisciplinary, socio-technical analysis of soft. ecosystem health," *arXiv:1711.04532*, 2017.

[16] K. Carillo, J. Marsan, and B. Negoita, "Exploring the biosphere—Towards a conceptualization of peer production communities as living organisms," in *Proc. AIS SIGOPEN Developmental Workshop for Research on Open Phenomena*, 2017, pp. 1–12.

[17] J. Dijkers, R. Sincic, N. Wasankhasit, and S. Jansen, "Exploring the effect of soft ecosystem health on the financial performance of the open source companies," in *Int. Workshop on Soft. Health*, 2018, pp. 48–55.

[18] D. Alami, M. Rodríguez, and S. Jansen, "Relating health to platform success: exploring three e-commerce ecosystems," in *European Conf. on Software Architecture Workshops*, 2015, pp. 1–6.

[19] J. Axelsson and M. Skoglund, "Quality assurance in software ecosystems: A systematic literature mapping and research agenda," *Journal of Systems and Software*, vol. 114, pp. 69–81, 2016.

[20] S. Hyrynsalmi, M. Seppänen, T. Nokkala, A. Suominen, and A. Järvi, "Wealthy, Healthy and/or Happy—What does 'ecosystem health'stand for?" in *Int. Conf. of Software Business*. Springer, 2015, pp. 272–287.

[21] K. Manikas and K. M. Hansen, "Reviewing the health of software ecosystems-a conceptual framework proposal," in *Int. workshop on software ecosystems*, 2013, pp. 33–44.

[22] S. da Silva Amorim, F. S. S. Neto, J. D. McGregor, E. S. de Almeida, and C. von Flach G. Chavez, "How has the health of software ecosystems been evaluated? A systematic review," in *Brazilian symposium on Soft. Eng.*, 2017, pp. 14–23.

[23] K. Blincoe, F. Harrison, and D. Damian, "Ecosystems in GitHub and a Method for Ecosystem Identification Using Reference Coupling," in *Working Conf. on Mining Software Repositories*, 2015, pp. 202–211.

[24] Z. Liao, N. Wang, S. Liu, Y. Zhang, H. Liu, and Q. Zhang, "Identification-Method Research for Open-Source Software Ecosystems," *Symmetry*, vol. 11, no. 2, p. 182, 2019.

[25] E. Constantinou and T. Mens, "Social and technical evolution of software ecosystems: A case study of Rails," in *European Conf. on Software Architecture Workshops*, 2016, pp. 1–4.

[26] M. Claes, T. Mens, and P. Grosjean, "On the maintainability of CRAN packages," in *Conf. on Software Maintenance, Reengineering, and Reverse Engineering*, 2014, pp. 308–312.

[27] E. Constantinou and T. Mens, "Socio-Technical Evolution of the Ruby Ecosystem in GitHub," in *Int. Conf. on Software Analysis, Evolution and Reengineering*, 2017, pp. 34–44.

[28] D. Spadini, M. Aniche, and A. Bacchelli, "Pydriller: Python framework for mining software repositories," in *European Soft. Eng. Conf. and Symposium on the Foundations of Soft. Eng.*, 2018, pp. 908–911.

[29] John Dantoni, " Mapping out Bitcoin's Ecosystem," https://www.theblockcrypto.com/genesis/12941/mapping-out-bitcoins-ecosystem.

[30] H. Borges and M. T. Valente, "What's in a GitHub star? understanding repository starring practices in a social coding platform," *Journal of Systems and Software*, vol. 146, pp. 112–129, 2018.

[31] T. J. McCabe, "A complexity measure," *IEEE Transactions on software Engineering*, no. 4, pp. 308–320, 1976.

[32] H.-C. Shih and E.-R. Liu, "New quartile-based region merging algorithm for unsupervised image segmentation using color-alone feature," *Information Sciences*, vol. 342, pp. 24–36, 2016.

[33] P. B. De Laat, "Governance of OSS: state of the art," *Journal of Management & Governance*, vol. 11, no. 2, pp. 165–177, 2007.

[34] D. Wahyudin, K. Mustofa, A. Schatten, S. Biffl, and A. M. Tjoa, "Monitoring the "health" status of open source web-engineering projects," *Int. Journal of Web Information Systems*, 2007.

[35] E. Constantinou and T. Mens, "An empirical comparison of developer retention in the rubygems and npm soft. ecosystems," *Innovations in Systems and Soft. Eng.*, vol. 13, no. 2-3, pp. 101–115, 2017.

[36] B. Lin, G. Robles, and A. Serebrenik, "Developer turnover in global, industrial open source projects: Insights from applying survival analysis," in *Int. Conf. on Global Software Engineering*, 2017, pp. 66–75.

[37] T. Mens, M. Claes, and P. Grosjean, "ECOS: Ecological studies of open source software ecosystems," in *Conf. on Software Maintenance, Reengineering, and Reverse Engineering*, 2014, pp. 403–406.

[38] V. Karakoidas, D. Mitropoulos, P. Louridas, G. Gousios, and D. Spinellis, "Generating the blueprints of the java ecosystem," in *Working Conference on Mining Software Repositories*, 2015, pp. 510–513.