Proceedings of 2018 1st IEEE International Conference on Hot Information-Centric Networking (HotICN 2018)

A Longitude Analysis on Bitcoin Issue Repository

Chelsea Hinds-Charles, Jenelee Adames, Ye Yang
School of Systems & Enterprises
Stevens Institute of Technology
Hoboken, New Jersey, USA
{chindsch, jadames, yyang4}@stevens.edu

Yusong Shen, Yong Wang
Department of Computer Science and Technology
Ocean University of China
Qingdao 266100, China
{shenyusong, wangyong}@ouc.edu.cn

***Abstract*—**As one of the most successful Blockchain systems, Bitcoin evolved over the past 8 years. The collaborative contribution of its online software development community gradually shaped the functionality and performance of Bitcoin. To date, most discussions around Bitcoin are from technologies underlying the product, as well as market applications. There are very few studies on the development and evolution processes of the Bitcoin software. It is important to investigate on such developmental issues, in order to better understand the development methodologies and lessons learnt from such a spearheaded Blockchain system. This paper serves this purpose by examining the issues data extracted from the Bitcoin GitHub repository from 2011 to 2018. It reports the trends of the major development issues from a longitude perspective. The main results include: 1) the average lifespan of an issue in Bitcoin issue repository is approximately 57 days; and 2) the Top-7 issue types including refactoring, tests, doc, RPC.REST.ZMQ, GUI, bugs, and wallet, accounting for 64.3% of all issues; 3) topic modeling techniques are beneficial in mining popularity and evolution of key issue topics and most problematic architecture components. Using data analysis and visualization techniques, this paper suggests the insights for significant development decisions such as better managing issue repository and strategic allocating of bug resolution effort.

Keywords—Bitcoin, Blockchain, Issues, GitHub Repository, Open Source, Software Engineering

I. Introduction

One of the most pivotal innovations of the 21st Century is the Bitcoin, also known as "digital gold". It is changing the way people think about money and how businesses operate all around the world. Currently, Bitcoin is being used for e-commerce, gaming, publishing, micro-transactions, and other online payments [5]. Bitcoin is referred to as a type of digital currency known as cryptocurrency. It operates on a decentralized peer-to-peer networked program without the need of a bank or a third-party regulator. The underlying technology behind Bitcoin is Blockchain. Blockchain is the digital ledger of transactions that exists as a shared and continually reconciled database [5]. The Blockchain database or network is not stored in any single location, meaning the records it keeps are truly transparent, easily verifiable and unalterable. Bitcoin has publicized Blockchain to extreme lengths and it is important to gain as much knowledge and understanding of the future of Blockchain because it will influence banks, purchases, supply chain management, and the global markets. All this will have an impact on regulations and transparency within businesses relationships.

978-1-5386-4870-4/18/$31.00 ©2018 IEEE

In efforts of learning the development trends, a clear understanding of the development process is necessary. Bitcoin development is agile. The development team uses the web-based open-source hosting software as the development repository called GitHub. This allowed for continuous system builds, continuous integration, and greater productivity. The repository development team included a distribution of the developers who were either contributors, members, owners and in some cases have no ties to the repository. Through GitHub, the development team tracked development tasks and enhancements through the issues functionality. With a development team spanning time zones and languages, the issues tracking covered all of the repository communications.

With Bitcoin becoming widely used across industries, it is beneficial to understand the development challenges of the product. Therefore, the problem at hand is to identify the prevalent issues within the system development, to gain insights of the trends of the prevalent issues within the Bitcoin infrastructure, and to know if there are any correlation between the bitcoin development and the bitcoin stock prices.

II. Research Design

A. Problem Statement

With Bitcoin becoming widely used across industries, it would be in our benefit to better understand the development process and challenges of the product. Therefore, the problem at hand is to understand the issues conveyed in the Bitcoin system GitHub repository, to gain insights on the evolution of Bitcoin infrastructure and features, in order to identify popular trends, and predict future research directions to improve the product quality, as well as technology market predictions.

B. Goal-Question-Metric Framework

The Goal Question Metric (GQM) approach is adopted to formulate research questions and necessary data metrics [3]. It consists of three parts. First, the target goals for the analysis need to be identified. In this case, the customer is anyone interested in learning about Bitcoin and Blockchain technology. Once the goals have been established, the second part is to determine questions that will help characterize the achievement of the goals. And finally, define the metrics that will provide a quantitative answer to each question.

The goal in this study is to understand the challenges, issues and development changes of Bitcoin from a longitude perspective. For this purposes of this paper, the focused

remained on high-level repository metrics and metrics on a component basis. For the selected components, the goal was to understand the characteristics of the issues which would give insight on the development process of the respective architectural component. Next, various questions were determined to develop the metrics. For example, one question which was determined to characterize the development of the system of systems is: What is the average lifespan of an issue within the repository? The same question was applied on a component level, such as: What is the average lifespan of an issue within the respective selected components?

The final step was coming up with metrics to answer each question of the model. The metrics results are discussed in the empirical results, with the component metrics details in Table 3. However, the full proposed metric details of each goal are omitted due to the space limit.

C. Bitcoin Issue Dataset

The dataset used for this project was the issues from the Bitcoin GitHub repository. The issues included open and closed issues from 2011 through to 2018. In order to collect the issue data, the team used the GitHub API. After obtaining the data in the appropriate excel format, the data scrub had to be completed before the data analysis. The parsed data had the dimensions of 120 columns with 1000s of records. The data was cleaned to 15 columns and 7081 records. One limitation with the dataset is that approximately 200 issues were not included from year 2014 due to GitHub API restrictions during the data pull.

In this dataset, some key issues tag words include: a) Refactoring - referring to a technique for restructuring an existing body of code to support future development and optimize code readability by altering its internal structure without changing its external behavior [4], b) Bug - an error in a Bitcoin development program [4], c) Feature - a unit of functionality of a software system that satisfies a requirement, represents a design decision, and provides a potential configuration option, d) Test - referring to the tasks noted in the repository to add test cases to the testing plan, or task to address unsuccessful tests, e) Documentation (doc) - referring to the tasks of documenting functionality, processes, knowledge sharing and other aspects of development.

D. Analysis Steps

Since the research motivation was driven on understanding the development status, the GQM framework was used to design research questions and metrics. Analytics on the discrete dimensions within the dataset were conducted. From this step of the research, interesting metrics surfaced, such as: there is a 92.4% refactor rate from refactor request to refactor completion, the average lifespan of an issue within the repository is approximately 57 days, there is a 93.5% bug removal rate, and the average development life cycle of a new feature issue is 328 days.

The analyses continued using Tableau for data visualization and knowledge discovering for empirical results to address the problem statement. More specifically, three perspectives were investigated: 1) the overall issue distribution profile in the repository; 2) the top issue trend analysis; 3) semantic analysis of top issue types. The next section will present the main empirical results from the study.

III. Empirical Results

A. Understanding the issue distribution within the repository

Figure 1 illustrates the summary of the number of open and closed issues in each year across the period of 2010-2018. As expected, there is a substantial amount of more closed ticket in each year. Based on the metric analysis, the following overall profiles have been observed:

- The average lifespan of an issue within the repository is approximately 57 days;
- There is an 87.5% refactor addressed rate for those proposed refactor request;
- There is a 93.5% bug removal rate; and
- The average development life cycle of a new feature issue is 328 days.

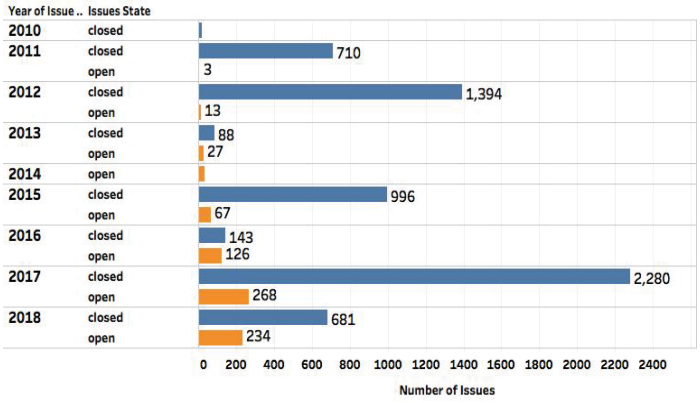

Fig.1. Number of Open and Closed Issues by Created Year

A compelling bug resolution is highlighted in the year 2017, with the highest closed issues of 2,280. This correlates with the time when Bitcoin achieved its peak share price of approximately $19,343 [8]. The indication is that the testing and bug resolution effort in 2017 significantly improved the functionality and performance of Bitcoin system, the more functioning and reliable Bitcoin system strived to great financial heights.

Figure 2 displays the most popular issue labels of the total Bitcoin repository issues. The Top-7 issue labels account for 64.3% of all issues, including: 1) *Refactoring* for changes related to code moving, code style fix, and code refactoring to address evolution needs; *2) Tests* for changes to the bitcoin unit tests or QA tests; *3) Docs* for changes to the documentation; 4) *RPC.REST.ZMQ* for changes to the RPC, REST or ZMQ middleware; *5) GUI* for changes to the bitcoin graphical user interface, e.g. bitcoin desktop wallet QT; *6)*

Bug for changes related to bug reporting and tracking; *and 7) Wallet* for the development of the Bitcoin wallet feature.

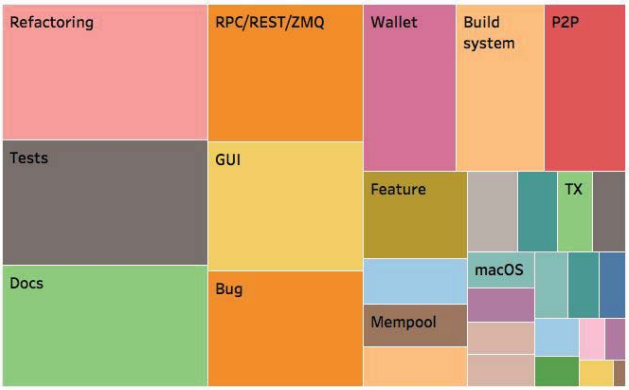

Fig. 2. Issues Label Distribution of All Issues

Table 1 summarizes the distribution of different issue types in Bitcoin repository. Other issues include more specific areas of concerns such as build systems, P2P, new feature, mempool, scripts and tools, validation, consensus, mining, etc.

Table 1. Summary of distribution of top issue labels

Issue Type Label	Number of Issues	%
Refactoring	528	0.11
Tests	488	0.10
Docs	473	0.10
RPC/REST/ZMQ	402	0.08
GUI	381	0.08
Bug	338	0.07
Wallet	293	0.06
Build system	277	0.06
Others	1611	0.34
Total	4791	1.00

Among the issue labels, *Refactoring*, *Bugs*, *Tests*, and *Docs* are some essential software engineering activities for, representing the application of specific methods, tools in order to correct code/design, verify correctness of functionalities, and document important information about the development processes. One surprising observation is that, as shown in Table 1, the Docs (i.e. documentation) issues consists of about 10% of all Bitcoin issues, which is large portion of the development effort, concerning that Bitcoin is development following agile development and/or open source software development processes. This indicates that for complex open/free software like Blockchain, the high priority on documentation might be one of the success critical factor to enable effective and consistent communication and exchange among diverse, cross-disciplinary stakeholders.

B. Top issues trend analysis

To understand the evolution of Bitcoin issue distribution, we further identified the Top-7 issue labels in each year, as shown in Figure 3. As expected, many of the most prevalent issues in each year of development are the issues previously identified.

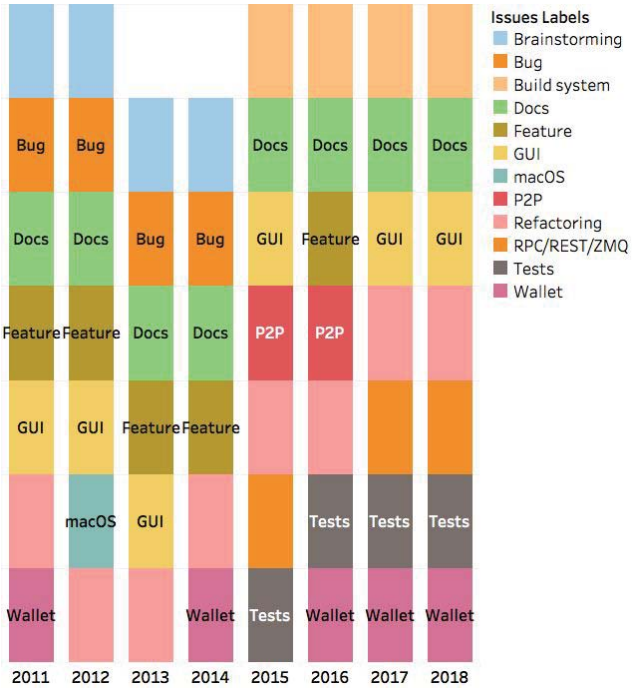

Fig. 3. Top 7 issues by year during the development lifecycle

More specifically, the most notable findings are: 1) During early stage of Bitcoin development (i.e. 2011-2014), more issues were discussed around *brainstorming*, *documentation*, *feature*, and *GUI;* 2) During middle stage (i.e. 2015-2016), *refactoring and tests* are placing important roles during the development process, more discussion focused on *Build system* and *P2P*; and 3) During more recent stage (i.e. 2017-2018), *RPC.REST.ZMQ* caught more attention than *P2P*, since Bitcoin 0.12.0 release introduced major changes of Random-cookie RPC authentication and notification through ZMQ. These results provide traces for reverse engineering the software development processes.

Fig 4 takes the seven selected issue tags and reveals the trend analysis of the issues quantities. Based on the trend data, it is shown that the number of issues for documentation and bugs are just as prevalent from 2015 onwards. The number of closed feature and GUI issues had a nominal variance throughout all of the years. In addition, there is a spike in the total number of the number of closed issues for all issue types. This can be due to the dataset flaw of minimal missing data from 2014. The results of the testing issues also showed 2015 to be a monumental year due to continuous testing being incorporated into the repository. The initial expectation was that tests would have more issues than refactoring since there are more test cases to check against compared to the need to

refactor. Another expected outcome is the number of bug issues decreasing as the number of test issues increases. Interestingly, the amount of both the bug issues and the test issues increase. However, the lower amount of test issues can be attributed to the developers using a continuous integration tool, i.e. Travis CI.

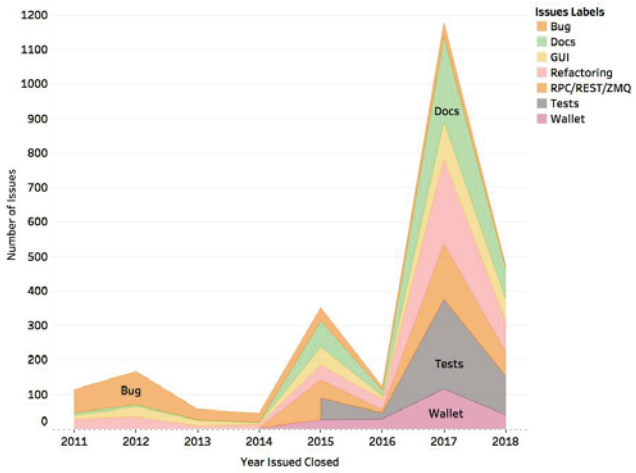

Fig. 4. Top-7 issues trend analysis

C. Semantic Analysis of Top Issue Types

1) Top-5 Topics for Top-3 Issue Types

A follow-up topic modeling analysis is conducted on the top issue types, as discussed in previous subsection A, to gain understanding of the semantic characteristics of issue descriptions. We applied the NLTK [1] library to pre-processing and clean the issue descriptions, and then conducted Latent Dirichlet Allocation (LDA) analysis using the gensim library [2].

For each top issue top, we identified the top 5 topics, each consisting of 5 topic words. For space saving consideration, we only show the results of topic modeling analysis on the Top-3 issue types (i.e. refactoring, tests, and doc), as summarized in Table 2.

Table 2. Illustration of the Top-5 topics

	Topic#	Topic Word-1	Topic Word-2	Topic Word-3	Topic Word-4	Topic Word-5
Refactoring	0	remove	commits	unsigned	linux	value
	1	class	refresh	script	include	header
	2	version	time	added	connection	case
	3	string	error	peer	time	see
	4	compare	remove	stripped	warning	static
Tests	0	directory	job	named	temp	unit
	1	line	functional	self	error	linux
	2	true	check	failure	running	functional
	3	master	line	self	core	pruning
	4	directory	key	time	check	commits
Doc	0	path	bin	line	added	release
	1	script	update	document	option	style
	2	lock	comment	time	example	peer
	3	debug	command	remove	text	value
	4	developer	think	work	tool	bit

Some topics draw our attention immediately, as we are manually checking the issue description text. For example, Topic0 in the "Refactoring" type is a vector of <remove, commits, unsigned, linux, value>, this corresponds to a set of refactoring to introduce "*an unsigned char which is a natural/direct representation for values"* ranging 0-16, and remove the need for 2 casts (i.e. BIP 141 and DecodeOP_N) in script/standard.cpp. Topic2 in the "Refactoring" type is the vector of <version, time, added, connection, case> indicates the discussion on handling a special case where a peer is sending obviously wrong information, and "*the big idea is to punish it by maybe dropping your connection (after certain time period) to it, and ban it's IP address so it cannot immediately re-connect*".

Topic0 in the "Tests" category corresponds to the test assurance of backup certain test files in temp path, so as not to accidentally "*overwrite a random file* with the same name *that happens to be in the current directory*". Other topics in this category capture other testing-oriented keywords including functional, error, failure, running, check, etc.

Unlike the first two categories, the "Doc" category consists of works such as script, comment, example, developer, think which are not directly representing the code itself. However, these words form more descriptive information regarding the intermediate development artifacts and programming context. and are critical for creating shared understanding, problem solving, and long-term maintenance, which are the critical success factor for such complex, dynamic collaborative development projects as Bitcoin.

Although these semantic analysis results are preliminary, the extraction of the most differentiating topics in each issue category is very important and critical for developing more predictive analytics to aid development decision making. For example, such topics can be used to train predictors for categorizing the most defective modules/files, or the more defect-introducing commits in the Bitcoin software; or for recommending the most relevant developers who have been working on the most similar bugs to fix a newly opened bug, and so on.

2) Issue Distribution across Bitcoin Components

The Bitcoin architecture can be viewed as the Bitcoin Financial System of System [6]. The foundation of the Bitcoin Network payment processors within this SoS focuses on four key concepts. The four concepts include: 1) Transactions, 2) Blocks/Blockchain, 3) Proof of Work, and 4) Protocol [7]. Fig 5 depicts the components which emulates the foundation of the Blocks/Blockchain concept foundation.

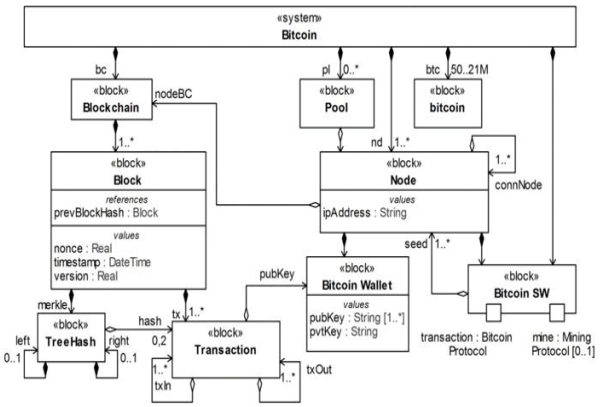

Fig. 5. Blockchain Data Model Design for Bitcoin [7]

We first identify the main component constructs from the above Blockchain Data Model (e.g. Blockchain, Block, Bitcoin Wallet, Transaction, Pool, Node, and Bitcoin, etc.), and then derive the top-5 topics from issues including these keywords. Then, we use these keywords to sort out the selected issue sets into ten subsets according to the above ten types. Finally, the heat map shows the reported issues count of the top ten issues with respect to the various Bitcoin architecture components. Figure 6 illustrates the resultant heat map.

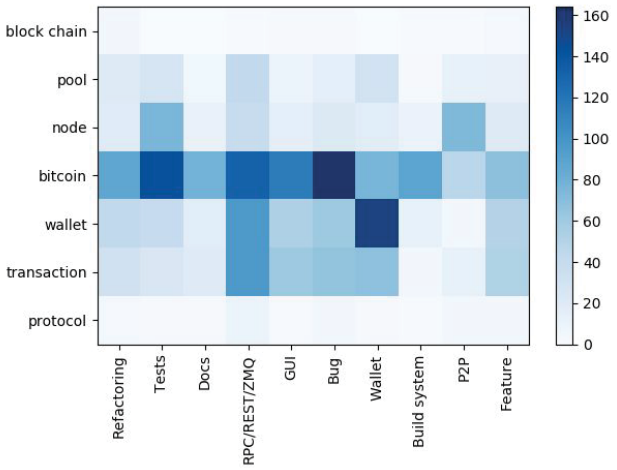

Fig. 6. Bitcoin component to top ten issues heat map

It is shown that there are some high issues quantities for the architecture components bitcoin, wallet, transaction, protocol, and node. Some of the popular issue types for these components include; tests, docs, RPC/REST/ZMQ, GUI, bug, and wallet. Fig 6 describes the following results: (1) Most of the issues in the development of bitcoin have been related to itself in the past 9 years. (2) Wallet development in the bitcoin also highlights a number of issues. (3) The middleware (e.g. RPC, REST and ZMQ) part of the bitcoin has shown many issues in various aspects in the past. (4) The protocol part of the bitcoin is very stable and there are not many issues. Additional empirical results on component level metrics are derived, as shown in Table 3. These results summarize the diverse characteristics of issue density, issue open duration, issue resolution rate, and dominant issue types associated with different components. It indicates that the component level issue has different influences on the development productivity and product quality.

Table 3. Bitcoin Component Metrics Conclusion Summary

	Blockchain	Pool	Node	Bitcoin
# of Issues	565	227	137	648
Average Length of Issues	95 days	50 days	55 days	78 days
Issues Resolution Rate	85.3%	85.9%	81.7%	89.5%
Top-3 issue types	RPC/REST /ZMQ, P2P, Refactoring	Mempool, RPC/REST /ZMQ, Wallet	Tests, Refactoring , P2P	Bug, Tests, GUI

IV. Discussion

From the overall repository and the component analytics results, the following conclusions can be drawn:

1. The relatively high refactor rate and refactor addressed rate from the overall repository analytics tells us that Bitcoin crypto technology underwent major design and architecture transformation to support the development of the platform. With the shortest issues life cycle of the three issue types and the most pull request percentage, refactoring is highly regarded within the development process because new functionality is highly regarded. However, looking at the analytics from a component point of view, refactoring issues played a major role for the Blockchain and the node architecture component development.

2. The relatively high bug removal rate in the repository tells us that Bitcoin crypto technology is proactive in addressing bugs to better understand and to overcome the Bitcoin financial system of systems development challenges. However, the component analytics confirms that the bug issues were prevalent in the Bitcoin component development. With the other components of the Bitcoin architecture being standard components for other Blockchain products, it is expected that bug and test issues would be prevalent within the bitcoin component development.

Some challenges in this study resulted in limitations and threats to the resulting findings. Because of data acquisition restrictions, the dataset excludes approximately 3% of the total issues from the Bitcoin repository. Even though this may diminish the accuracy of the value of some findings, there is still some validity to the proportions of the metric results. Secondly, the limited amount of sources discussing the Bitcoin architecture limited our architecture knowledge to be dependent on one source. Another threat to the validity of the

findings is due to the inconsistency with the issues tags, description and issues titles of a number of issues within the repository. Whiles completing the development analytics, we noticed that a possible source of error is incorrect issues labeling and issues which were not labeled. One major lesson that came out of this research project is the importance of digging deeper for true knowledge discovery.

V. Conclusion and Future Work

The exploration of Bitcoin began with understanding the Blockchain technology. That led to understanding the development of the Bitcoin cryptocurrency through GitHub. This paper reported the trends of the major development issues from a longitude perspective. The main results include: 1) the average lifespan of an issue in Bitcoin issue repository is approximately 57 days; and 2) the Top-7 issue types including refactoring, tests, doc, RPC.REST.ZMQ, GUI, bugs, and wallet, accounting for 64.3% of all issues; 3) topic modeling techniques are beneficial in mining popularity and evolution of key issue topics and most problematic architecture components. Using data analysis and visualization techniques, this paper suggests the insights for significant development decisions such as better managing issue repository and strategic allocating of bug resolution effort.

A next step in this research would be to develop predictive analytics to improve the effectiveness of issue resolution process of the Bitcoin development, and to predict features/issues/refactors that benefit the maximization of financial value of bitcoin. This paper anchored what can be further development on as Bitcoin development analytics to understand the challenges and the development process of the Bitcoin software.

References.

[1] NLTK for Natural language processing: http://www.nltk.org/.
[2] Gensim library: https://pypi.python.org/pypi/gensim.
[3] Laird, L. M., & Brennan, M. C. (2006). Software measurement and estimation: A practical approach. Hoboken, NJ: John Wiley & Sons. p. 9
[4] Van Solingen, Rini; Egon Berghout (1999). The Goal/Question/Metric Method. McGraw-Hill Education.
[5] Franco, P. (2014). Understanding Bitcoin: Cryptography, Engineering and Economics.
[6] Nakamoto, S. (2017). Bitcoin: A Peer-to-Peer Electronic Cash System. https://www.bitcoin.com/info/what-is-bitcoin-cash
[7] Roth, S. (2015). An Architectural Assessment of Bitcoin Using the System Modeling Language.
[8] Coindesk price website: http://www.coindesk.com/price/