Empirical Software Engineering (2019) 24:2636–2673
https://doi.org/10.1007/s10664-019-09708-7

Understanding the motivations, challenges and needs of Blockchain software developers: a survey

Amiangshu Bosu[1] · Anindya Iqbal[2] · Rifat Shahriyar[2] · Partha Chakraborty[2]

Published online: 27 April 2019

Abstract

The blockchain technology has potential applications in various areas such as smart-contracts, Internet of Things (IoT), land registry, supply chain management, storing medical data, and identity management. Although GitHub currently hosts more than six thousand active Blockchain software (BCS) projects, few software engineering researchers have investigated these projects and their contributors. Although the number of BCS projects is growing rapidly, the motivations, challenges, and needs of BCS developers remain a puzzle. Therefore, the primary objective of this study is *to understand the motivations, challenges, and needs of BCS developers and analyze the differences between BCS and non-BCS development*. On this goal, we sent an online survey to 1,604 active BCS developers identified by mining the GitHub repositories of 145 popular BCS projects. The survey received 156 responses that met our criteria for analysis. The results suggest that the majority of the BCS developers are experienced in non-BCS development and are primarily motivated by the ideology of creating a decentralized financial system. Although most of the BCS projects are Open Source Software (OSS) projects by nature, more than 93% of our respondents found BCS development somewhat different from a non-BCS development as BCS projects have higher emphasis on security and reliability than most of the non-BCS projects. Other differences include: higher costs of defects, decentralized and hostile environment, technological complexity, and difficulty in upgrading the software after release. These differences were also the primary sources of challenges to them. Software development tools that are tuned for non-BCS development are inadequate for BCS and the ecosystem needs an array of new or improved tools, such as: customized IDE for BCS development tasks, debuggers for smart-contracts, testing support, easily deployable simulators, and BCS domain specific design notations.

Keywords Blockchain · Cryptocurrency · Survey · Bitcoin · Ethereum · Motivation · Challenges

Communicated by: Christoph Treude

✉ Amiangshu Bosu
amiangshu.bosu@wayne.edu

Extended author information available on the last page of the article.

1 Introduction

In 2008, a person or entity under the pseudonym *Satoshi Nakamoto*, published a whitepaper to introduce 'Bitcoin,' a digital currency (aka cryptocurrency) based on an immutable and decentralized public ledger known as *blockchain* (Nakamoto 2008). Currently, 'Bitcoin' is the leading cryptocurrency with a market cap of over 100 billion USD. On the other hand, the blockchain technology that runs Bitcoin could prove to be much more significant (Swan 2015), as a blockchain can record transactions between two parties efficiently and in a verifiable and permanent way; thus eliminating the need for the third party in the middle. Moreover, the availability of all the transactions ever completed to all nodes makes a blockchain-based system more transparent than centralized solutions. Therefore, apart from its application in cryptocurrency, the blockchain technology has potential applications in various other domains such as smart-contracts (Delmolino et al. 2016), Internet of Things (IoT) (Huh et al. 2017), land registry (Underwood 2016), supply chain management (Korpela et al. 2017), storing medical data (Azaria et al. 2016), and identity management (Jacobovitz 2016).

The technological innovation and fundamental changes required in the design, development, and deployment of blockchain have also attracted tremendous interests from the software development community. For example, a recent study (Chakraborty et al. 2018) from March 2018 reported 3,000 Blockchain software (BCS) projects hosted on Github.[1] In October 2018, within seven months, that number stands at 6,800 (i.e., more than doubled). Unlike traditional development, blockchain developers need to be cautious about malicious actors, secure an immutable and distributed database, and design efficient and reliable protocols withstanding the scarcity of tools and resources which is unavoidable for a new technology. The unalterable nature of a blockchain makes the recovery of an error prohibitively difficult or practically next to impossible if the vulnerability is detected after the deployment. Although some other large-scale software such as financial applications requires similar robustness, the rapidly changing blockchain ecosystem adds significant new challenges (Porru et al. 2017). The significant differences between traditional software development (i.e., non-BCS) and blockchain oriented development motivated (Destefanis et al. 2018) even to propose a new development paradigm named Blockchain-Oriented Software Engineering (BOSE).

Although several characteristics of the BCS technology suggest BCS development to be different from a non-BCS development, very few software engineering (SE) research has focused on the former. Therefore, several questions regarding BCS development still remain a puzzle. For example, i) *Who are the BCS developers and what are their motivations behind joining BCS development?* ii) *Is BCS development indeed different from a non-BCS development?* iii) If different, *what are the areas that mark those differences?* iv) Do current development tools and practices that are tuned for non-BCS development satisfy the needs of BCS developers? v) If not, *what are the tools and techniques that BCS developers are in need?*

Answering these questions would enable the research community and the providers of technical tools to build necessary support to design, develop, test, and deploy BCS applications. Moreover, developers interested to join BCS projects are likely to have valuable guidance for their preparation. Therefore, this study aims *to understand the motivations, challenges, and needs of BCS developers and analyze the differences between BCS and*

[1] https://github.com/topics/blockchain

non-BCS development. We also aim to conduct a comparative analysis of BCS with non-BCS development to pinpoint specialized tasks that may benefit the former.

On these goals, we designed and sent an online survey to 1,604 BCS developers gathered by mining the GitHub repositories of 145 BCS projects. A survey is an ideal instrument for this study as current BCS developers have first-hand experiences of their challenges and needs. The survey received 156 responses from BCS developers that met our criteria for analysis. We adopted a systematic qualitative analysis approach to build a coding scheme for the open-ended responses. Using a qualitative analysis software, multiple coders independently assigned codes to each response and achieved a 'substantial' inter-rater reliability (Cohen's $\kappa = 0.62$) (Cohen 1960). We reported partial results of the survey in a recent publication (Chakraborty et al. 2018), which explored only the software development practices (i.e., verification and validation, task assignment, requirement analysis, and communication and collaboration) of BCS projects. On the other hand, this publication focuses on a different set of questions none of which was included in our prior article.

The primary contributions of this study include:

- A better understanding of BCS developers' motivations, challenges, and needs;
- A comparative analysis of BCS with non-BCS development;
- A comparative analysis of the results of our survey with two prior SE surveys to set our results into the perspective of the software development realm.
- Potential directions for SE researchers to build supports for BCS development; and
- A characterization of the contemporary BCS development community.

The remainder of the paper is organized as follows. Section 2 provides a background on blockchain and a overview of a prior motivating study. Section 3 introduces the research questions of this study. Section 4 describes our research methodology. Section 5 describes the demographics of our respondents. Section 6 presents the results of this study. Section 7 discusses the implications of our findings. Section 8 describes the threats to the validity of our results. Finally, Section 9 concludes the paper.

2 Background

This section provides a brief overview of the key concepts on blockchain and a prior SE study that assisted in designing our survey questions to compare BCS development against non-BCS.

2.1 Blockchain

`Blockchain` is a decentralized, peer-to-peer, transparent, immutable, and append-only data storage. It keeps a permanent record of writes called `transactions`. Multiple transactions are grouped in `blocks`. Each block in a blockchain contains its hash computed using a well-known hashing or proof-of-work (Jakobsson and Juels 1999) algorithm (e.g., SHA256, ethash, and equihash) and the hash of the previous block called parent block (Fig. 1). The first block in a chain is called the `genesis block`, which does not have any parent. Each block's hash is calculated based on its data, current timestamp and the hash of its parent block. Any change in a block's data causes alteration of its hash and invalidates all the subsequent blocks, and the tampering becomes immediately evident to every member node of the chain. Hence, to compromise a blockchain, collusion of the

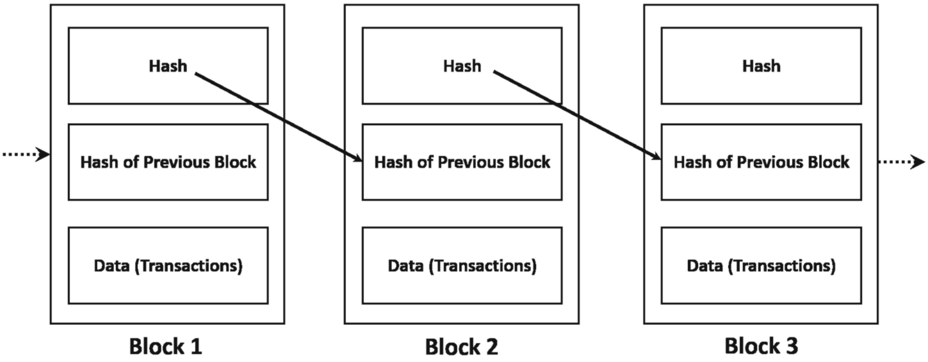


Fig. 1 Simplified diagram of a blockchain

majority of the network is required which is impractical in case of a large blockchain (Narayanan et al. 2016). Therefore, blockchain is a chain of blocks where the blocks are irreversible and immutable.

All the nodes in a blockchain network participate simultaneously in finding the next block to write. This process is called `mining`, where the nodes calculate a hash value by adding a nonce (i.e., a random value) to a list of transactions waiting to be added to the blockchain. To be eligible as the next block, the hash must be smaller than an agreed-upon value (known as `difficulty`), and the nodes continue calculations using different nonces until they find a nonce that generates a hash satisfying the 'difficulty.' The node finding a new block will broadcast it to all other nodes in the network to confirm the correctness of this new block. Once confirmed, the new block is added to the blockchain, and each of the transactions contained in the block is considered `verified` (Chuen 2015). The finder is usually rewarded with a pre-defined number of tokens, known as `block reward`. The difficulty of the next block is determined by the network with a pre-defined algorithm.

There is no central control over the operation of a blockchain. The underlying philosophy is that no single participant or group of participants can control the infrastructure and all the participants in the network have an equal role to play. In the absence of a central controller, the transactions are mediated by the member nodes using a `consensus` protocol, which ensures that all the nodes have an identical copy of the blockchain. A new block is considered verified only after the majority of the member nodes vote it as true and trustworthy using the consensus protocol. A blockchain's security is based on the assumption that tampering would have to happen across the majority of the nodes (aka `51% attack`) of a network in the same way simultaneously. So once a blockchain network achieves critical mass, altering a blockchain posthoc becomes infeasible.

In the context of the blockchain, public key cryptography (Garfinkel 1996) ensures the integrity and authenticity of any message/transaction. Each node owns a pair of asymmetric encryption keys (Simmons 1979), where the `public key` is broadcast to all relevant nodes but the `private key` is kept secret. A sender signs messages with its own private key and a receiver verifies the integrity of the message by decrypting it with the sender's public key. In cryptocurrency applications, the public key of a user also acts as his/her `account address`. Therefore, a user must sign outgoing transactions using his/her private key. A miner node would verify an outgoing transaction from an account only when it can authenticate the transaction using the owner's public key.

One of the recent innovative applications of blockchain is `Smart-contracts`, which are self-executing contracts with the terms of the agreement between buyer(s) and seller(s) of transactions written using lines of code instead of a legal language. Smart-contracts permit trusted transactions and agreements to be carried out among different anonymous parties without the need for a central authority, legal system, or external enforcement mechanism. Since a smart-contract, once deployed, lives on a distributed and decentralized blockchain network, it remains traceable, transparent, and irreversible.

2.2 Game Development vs. Traditional Development at Microsoft

One of the objective of this study is to identify the differences between BCS and non-BCS development. The design of our survey questions to investigate this objective is motivated by a prior study at Microsoft (referred as 'MS study' hereinafter) by Murphy-Hill et al. (2014). The primary objective of the MS study was to compare game development against traditional software development. On this goal, the MS study interviewed 14 developers who had both game and non-game development experiences. Those interviews focused on topics from the 10 areas in the Software Engineering Body of Knowledge SWEBOK (Abran et al. 2004) as well as general work features from applied psychology (Humphrey et al. 2007). After analyzing the interview transcripts, the authors selected 28 statements to assess the differences between game and non-game development at Microsoft.

In a survey of Microsoft developers from three different domains (i.e., Game, Office, and Other), each of the respondents were asked to rate each of the selected 28 statements on a 5-point Likert scale from 'Strongly Disagree' to 'Strongly Agree'. The results of the survey identified differences between game and non-game development in terms of i) having clear requirements, ii) using agile development methods, iii) valuing creativity, iv) communicating with non-engineers, v) team compositions, and vi) taking pride for developed software. The empirically developed 28 statements spanning various software engineering as well as general work feature is another key contribution from this study, since these statements can be reused to compare the opinions of software developers from two different domains.

3 Research Questions

The primary objective of this study is *to understand the motivations, challenges, and needs of BCS developers and analyze the differences between BCS and non-BCS development.* We aim to achieve this goal based on four specific research questions. We also explore one additional research question to characterize the contemporary BCS development community. Following subsections introduce the five research questions with a brief motivation behind each question.

3.1 Personal Characteristics of the BCS Developers

Characterizing the Open Source Software (OSS) developers has drawn interests from the researchers (David and Shapiro 2008; Dahlander and Mckelvey 2005; Hars and Ou 2001; Lakhani and Wolf 2003; Mockus et al. 2002) to understand the distribution of expertise, the strength (and durations) of their attachments to particular projects, and the recruitment and retention of newcomers. However, such a characterization for BCS developers is currently missing. There is a common concept that blockchain development community is dominated by libertarian and anarchist groups who consider it as a means of removing control from an imposing authority (Galati 2019). The contributors like to create a system that regulates itself and provide advantages to those willing to take part in it (Reijers and Coeckelbergh 2018). Hence, it is worth formal study the demographics of the BCS developers regarding their age, gender, education, and general software development experience and see if it differs from non-BCS. Also, the future participants of BCS development are likely to have idea about the characteristics of existing community who are involved in the industry with large stake, wide variety of motivation and ethical consideration. Since this understanding has importance, our first research question is:

RQ1: *Who is contributing to BCS projects?*

3.2 Motivations of BCS Developers

Much research has focused on understanding the motivations of OSS developers (Hars and Ou 2001; Hertel et al. 2003; Lee et al. 2017; Roberts et al. 2006; Lakhani and Wolf 2003) and have identified five primary categories motives as following:

– *Intrinsic motivations* refer to a person's desire to feel competent and self-determined. Intrinsic motives are directly linked to the emotions of interest and enjoyment (Deci and Ryan 1985). Examples of such motives include fun, hobby, self-interest, and feeling competent.
– *External rewards* refer to direct or indirect incentives, which include monetary compensation, benefits through software usage, or prospects for career growth (Lakhani and Wolf 2003).
– *Ideology* includes the norms, beliefs, and values shared among the developers of an OSS project (Stewart and Gosain 2006). For example, the Free Software Foundation was established on the ideology of providing users with freedom to use, modify, and redistribute software (Stallman 2003).
– *Community recognition* corresponds to a person's needs for belonging and love. A developer desiring to identify him/herself as a member of an OSS community seeks that recognition from other community members (Hars and Ou 2001).
– *Learning* opportunities offered by OSS projects help increasing one's 'human capital' (i.e., personal skills, capabilities, and knowledge) by means of education, training, learning, and practicing. Since these human capital gains eventually leads to better job opportunities, higher salaries, and more fulfilling jobs, learning is one of the motives among many OSS developers, especially newcomers (Hars and Ou 2001).

The results of those studies found "intrinsic motivations" ranking top among OSS developers. Although most of the BCS projects are also OSS projects, large inflows of cash through ICO (Initial Coin Offerings) or token sales (Adhami et al. 2018), which are very common for BCS projects are rare for non-BCS OSS projects. Many of the early BCS developers / investors have garnered significant financial rewards from the recent boom of the cryptocurrency market. Therefore, it won't be surprising if external rewards are the primary motives for many of the BCS developers. Our next research question tries to find out whether BCS developers' motivations are similar to non-BCS OSS developers or they are attracted by potential financial gains. Understanding the motivations of BCS developers is important since it will help to identify prospective joiners, which may form synergies with a BCS community. Hence, our next research question is:

RQ2: *What are the primary motivations of BCS developers?*

3.3 Differences Between BCS and Non-BCS Development

Although BCS projects possess many traits of traditional OSS projects, we expect some differences between BCS and non-BCS development due to several characteristics (e.g., the immutability of data, hostile environment, and difficulty of upgrading the software after deployment) that distinguish the BCS domain from the non-BCS one. Since identifications of the differences between BCS and non-BCS development will allow assessing the applicability of various traditional SE tools and techniques for BCS projects, we seek to find out:

RQ3: *What are the differences between BCS and non-BCS development?*

3.4 Challenges of BCS Development

Most of the innovative applications of the blockchain technology such as smart-contracts and distributed applications (aka dapps) are at nascent stages. The BCS development landscape is also changing at a rapid pace with projects fiercely competing with each other to emerge as the market leader (Krafft et al. 2018). Although the high costs of bugs mandate high reliability from a BCS, developers face tremendous pressures from the investors to release the product. These scenarios coupled with the unique characteristics of the BCS domain pose several challenges to the BCS developers. We believe identifications of the primary challenges of BCS development will provide prospective joiners with guidance for their preparation and encourage research to mitigate those challenges. Hence our next research question is:

RQ4: *What are the primary challenges of BCS development?*

3.5 Tools that BCS Developers Need

The difficulty of testing BCS ranks among the top of challenges encountered by BCS developers (Porru et al. 2017; Brooke 2018). Current testing tools cannot simulate testbeds to simulate the distributed and hostile execution environment of a BCS. Moreover, smart-contract development, which is gaining popularity lacks supporting tools (Clack et al. 2016). The requirement of tools, once clearly understood, will lead to the development of supporting tools to design, develop, test, and deploy BCS applications. Since contemporary BCS developers have the first-hand knowledge of their needs for supporting tools, we inquire:

RQ5: *What are the tools that BCS developers currently need?*

4 Research Methodology

Since the five research questions of this study are geared towards gathering the opinions of BCS developers, we chose a survey as our research instrument. The remainder of this section describes the survey design, our participant selection criteria, pilot testing, data collection, and qualitative data analysis.

4.1 Survey

Our goal in designing the survey was to keep it as short as possible, while still gathering all of the relevant information. For the current paper, we only consider a subset of the survey questions, while questions focusing on software development practices of BCS projects were reported in a recent publication (Chakraborty et al. 2018). Table 1 lists each survey question included in this paper, the research question that motivated its inclusion, and the answer choices provided. Questions indicated with a ‘D,’ rather than a ‘RQ#’ were included to gather demographics about the respondents. For the questions that were open-ended, there are no specified answer choices. Several of the open-ended and demographics questions were inspired by previous surveys (Murphy-Hill et al. 2014; Hertel et al. 2003; Von Krogh et al. 2012; Hars and Ou 2001; Lakhani and Wolf 2003; West and Gallagher 2006).

To compare BCS with non-BCS development, the survey asked the respondents (Q13 in Table 1) to rate their agreements for 16 statements on a 5-point Likert scale from ‘Strongly disagree’ to ‘Strongly agree.’ These statements (‘Statement’ column in Table 3) were adopted from a prior SE survey at Microsoft (Section 2.2) to assess the differences between the game and non-game development (Murphy-Hill et al. 2014). While the MS study had 28 statements, our survey did not include the statements regarding the company or manager as BCS projects are mostly community driven.

4.2 Participant Selection

To ensure valid results, we only surveyed BCS developers with sufficient experience. We identified 145 BCS projects based on following four criteria:

- Tagged under at least one of the following six ‘topics’:[2] `blockchain`, `cryptocurrency`, `altcoin`, `ethereum`, `bitcoin`, and `smart-contracts`.
- ‘Starred’ by at least ten users.
- Have at least five distinct contributors.
- A manual verification of the repository confirmed it as a BCS project.

We used Github API[3] to identify 1,604 contributors, each of whom had submitted at least five changes to one of those 145 projects.

We mine the Git commit logs of the identified 145 projects to gather the email addresses of those 1,604 active contributors. We also got the survey questions, consent form, participant selection strategy, solicitation email, and data management reviewed and approved by our university’s Institutional Review Board (IRB).

4.3 Pilot Survey

To help ensure the understandability of the survey, we asked Computer Science professors and graduate students with experience in SE and experience in survey design to review the survey to ensure the questions were clear and complete. The feedback only suggested minor edits. The changes we made include: adding more answer choices to several questions and adding clarifying examples to three questions.

4.4 Data Collection

On December 13, 2017, we sent each of the 1,604 BCS developers in our list a personalized email mentioning the BCS repository that we mined to obtain his/her email address

[2] https://blog.github.com/2017-01-31-introducing-topics/
[3] https://developer.github.com/v3/

Table 1 Survey questions

#	RQ*	Question text	Answer choices
Q1	RQ1	How old are you?	[#]
Q2	RQ1	Which gender do you identify yourself with?	[Male, Female, Prefer not to disclose]
Q3	RQ1	What is your highest level of education?	[High school, Bachelors, Masters, Ph.D.]
Q4	RQ1	How many years of software development experiences do you have?	[Less than a year, between one to five years, between six to ten years, more than ten years]
Q5	D	How many years have you been developing blockchain software?	[Less than a year, between one to two years, between three to five years, more than five years]
Q6	D	What is your primary Blockchain software project (i.e. the project that you have spent most of your time)	[#]
Q7	D	Do you receive any direct compensation (i.e, salary or cryptocurrency) from your primary project?	[I get directly paid with a FIAT currency, I get compensated with shares, tokens, or cryptocurrency, I receive both FIAT salary and tokens/shares, No, I do not receive any direct compensation]
Q8	D	Approximately, how many pull requests have you submitted to your primary project?	[Less than 10, Between 11 to 30, More than 30]
Q9	D	Approximately, how many hours on average do you spend per week on your primary project?	[Less than 5, between 6 to 10, Between 11 to 20, Between 21 to 35, I work full time]
Q10	RQ2	What are your motivations to contribute to your primary project?	
Q11	RQ3	Based on your experiences, what are the primary differences between blockchain and non-blockchain software development ?	[#]
Q12	RQ3	*The following statements aim to compare blockchain and non-blockchain software development. For each of the following items please rate how you agree or disagree with that statement.* Total 16 statements (shown in Table 3)	[Strongly disagree, Disagree, Neutral, Agree, Strongly Agree]
Q13	RQ4	What are the most challenging aspects of blockchain software development?	
Q14	RQ5	Please describe the type of tools that you currently do not have, but if implemented, can greatly help your blockchain software development activities.	[#]

* *'RQ#' numbers refer to the research question that motivated the inclusion of the survey question.*

* *Questions indicated with a 'D', were included to gather demographics about the respondents.*

with a link to the survey hosted on Qualtrics (Snow and Mann 2013). We also asked the respondents through both the solicitation emails and a reminder in the survey to answer our questions based on his/her personal experiences with the BCS project where we obtained his/her email address. Since 62 of our solicitation emails bounced, we were left with at most 1,542 potential participants, assuming all other emails actually reached their intended recipient. On December 21, 2017, we sent a reminder email. We closed the survey on January 5, 2018; after the response rate slowed to almost no response each day.

Data from the survey link created with Google's URL shortener showed a total 358 clicks on the survey URL (≈23% of the invitations). Out of those clicks, 200 people took the survey with a response rate of ≈13% (200/1542). As most of the questions were optional, many respondents skipped some of the questions. Only 115 respondents answered all the questions. After the exclusion of the 44 responses that did not answer either at least 75% of the questions or at least one open-ended question, we were left with 156 responses for analysis.

We also collect the developer demographics data from the *2018 Stack Overflow Annual Developer Survey* (referred as the 'SO Survey' hereinafter) (Overflow 2017). StackOverflow has been running the annual developer survey since 2011. The primary objective of the SO Surveys are to learn who contemporary developers are and what they need. These surveys cover a wide range of developer demographics such as age, education, location, gender, role, experience, ethnicity, and favorite technologies. Since the '2018 SO survey' was responded by total 98,855 developers from 183 countries worldwide, it is an accurate overview of the active software developers worldwide. Therefore, a comparison against the demographics of the SO Survey will enable us to identify if the BCS community is underrepresented or overrepresented by certain groups.

After contacting the authors of the MS survey, we were able to obtain their dataset, which allows us to compare and contrast BCS development with three non-BCS domains (i.e., *Games*, *Office*, and *Other*) from Microsoft (MS).

4.5 Qualitative Analysis Process

For the open-ended questions, we followed a systematic qualitative data analysis process. First, two of the authors independently extracted the general themes from the first 75 responses to each question. Using those themes, the authors had discussion sessions to develop an agreed-upon coding scheme for each question. Using this coding scheme, another author went through the remaining answers to determine any additional codes that need to be added.

With this scheme, two of the authors independently coded each response using the Coding Analysis Toolkit (CAT) (Lu and Shulman 2008) software. The coders could also add new codes, if necessary. We computed the level of inter-rater reliability of the manual coding process using Cohen's kappa (Cohen 1960), which was measured as 0.62. While there is no universally accepted 'good' kappa, values between 0.61 to 0.80 are generally recognized as 'substantial agreements' (Landis and Koch 1977). We used CAT to identify the discrepancies in coding and had discussion sessions to resolve all conflicts. Once we completed the coding process, we transferred the data into IBM Corp. Released (2017) for further analysis along with the quantitative data.

As a result of the coding process, a large number of codes emerged from each of the open-ended questions. To help with our analysis, we had discussion sessions to identify the codes that express similar themes. We grouped those codes into a smaller number of high-level categories with each category including one or more codes expressing a similar theme.

Table 4 in the Appendix shows the codes that emerged from our open-coding of the four survey questions and the categories that we assign each code to for the four open-ended questions included in this paper.

5 Demographics

To provide a proper context for the results, this section describes the demographics of the projects represented by the respondents and of the respondents themselves.

5.1 Projects Represented

Table 2 provides the results to Q6 (Table 1) about respondents' primary projects. The number in parenthesis represents the number of respondents who listed that project. Our respondents represent 61 different BCS projects. The Coin Development Index (Gilbertson and Vroegindewey 2017), which tracks the top BCS projects, indicates our respondents representing 18 out of the top 25 projects. Also, 37% of our respondents coming from the top ten projects indicates the participation of the top BCS developers in our survey.

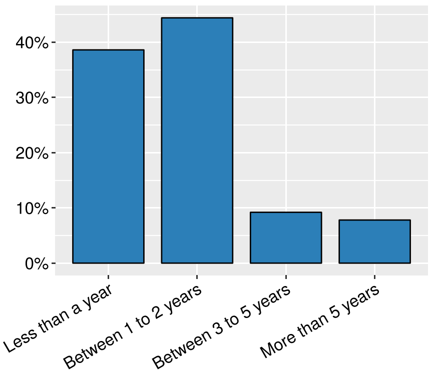
5.2 Respondents' Demographics

In response to Q5, 81.4% of our respondents indicated having less than 2 years of BCS development experiences, while 37.8% had less than a year (Fig. 2a). In terms of compensation (Fig. 2b), 43.1% of our respondents do not receive any direct financial benefits. The ratio of volunteers among our respondents is similar to the ratios reported in prior studies (Bosu et al. 2017; Bosu et al. 2014a). Among the respondents receiving direct financial compensations, 37.3% reported receiving shares, tokens or cryptocurrencies that may further motivate them to spend efforts to make their projects successful.

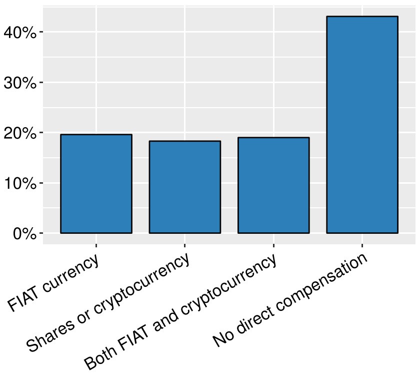
Table 2 Projects represented by our respondents

Multiple occurrences		Single occurrences		
Ethereum (22)	Bitcoin (9)	Ambisafe	Basic Identity Token	Bytom
Bitshares (7)	Monero (6)	Cpuminer	Dash	Distense
Sia (6)	Waves (5)	DNSChain	Ebets	ESKU
Solidity (5)	Lbry (4)	Etherbet	Etherplay	Fabric Labs
Ripple (4)	Nem (3)	Golem	Haskoin	ZeroLink
Cardano (3)	Decred (3)	Icofunding	Ind	Iroha
EOS (3)	Hyperledger (3)	JS Miner	Keyrun	Libsnark
IOTA (3)	Factom (2)	LiteCoin	Payroll System	PHP-Mpos
Feather coin (2)	Lisk (2)	Populus	Progmathon	Pycoin
Metamask (2)	Namecoin (2)	Shapeshift	Snapcoin	Status
Neo (2)	Remix IDE (2)	Steller	Storj	Swifty
Stratis (2)	Trezor (2)	Vandal	Vcash	Viper
Zcash (2)	Undisclosed /Private (14)			

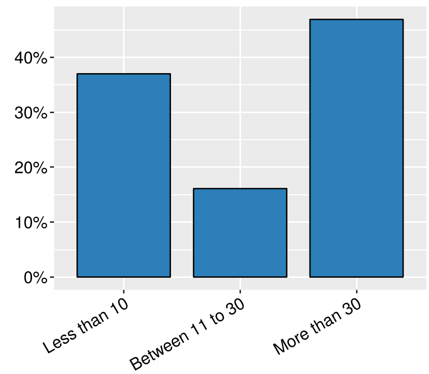
The numbers in parentheses under the columns 'Multiple occurrences' represent the number of respondents who listed that project. Only one respondent from each of the projects listed under 'Single occurrences'



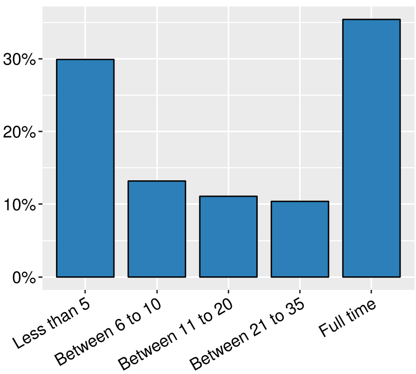
(a) BCS development experience



(b) Type of financial compensation



(c) Total number of code commits in BCS projects



(d) Average number of hours per week spent on BCS development

Fig. 2 Demographics of the respondents

In terms of the number of contributions to a BCS project (Fig. 2c) 57.6% of our respondents have made more than 10, while 42.9% had submitted more than 30. On the other hand, 42.7% of our respondents spend at least 20 hours a week on a BCS project, and 32.7% were working full time (Fig. 2d). Combining our respondents' number of commits and number of hours per week spent in BCS projects, we conclude that our respondents include a sample of active BCS developers from the top BCS projects who are qualified to provide valuable insights for the goals of this study.

6 Results

The following subsections describe the results of our survey by answering the five research questions introduced in Section 3. To help clarify the results, we also include excerpts from the qualitative responses to the open-ended questions. Each of the excerpts is followed by a number representing a unique identifier for the respondent who expressed that opinion. For example, [#5] indicates a response from respondent number 5. In a qualitative analysis, each open-ended response could match multiple codes. Therefore, the sum of the percentages can be greater than 100%. We conclude each of the subsections with a brief discussion of the key takeaways from the results of each research question.

6.1 RQ1: Who is Contributing to BCS?

Figure 3 shows the personal characteristics of the BCS developers in terms of their age (Q1), gender (Q2), education (Q3), and software development experience (Q5). We also compare these characteristics with the results from the that includes the demographics of 98,855 developers around the globe.

Around 95% of our respondents are males compared to only 3% females (Fig. 3a). These numbers suggest that the ratio of females in BCS development may be lower compared to the the software development community as reported in the SO survey (6.7%).

In terms age (Fig. 3b), our respondents are younger compared to the general software developer population. The distributions are noticeably different for two age groups. First, developers, who are aged between 26 to 30 years represent 31% of the BCS developers, compared to 25.4% general software developer population coming from the same age group. On the other hand, although 15.9% general software developer population are aged 40 years and higher, only 11.7% of the BCS developers belong to that age group. Therefore, BCS development may be attracting more young developers than the veterans.

In terms of the highest level of education (Fig. 3c), our respondents are more qualified with 15.4% high school graduate, 46.8% with a bachelors degree, 32.7% with a masters degree, and 5.1% with a Ph.D. The corresponding numbers in the SO survey are 23.4%, 47.7%, 23.2%, and 2.2% respectively. The higher educational qualifications of our respondents may not be surprising as BCS development requires more in-depth knowledge of computing than non-BCS development (Zheng et al. 2016).

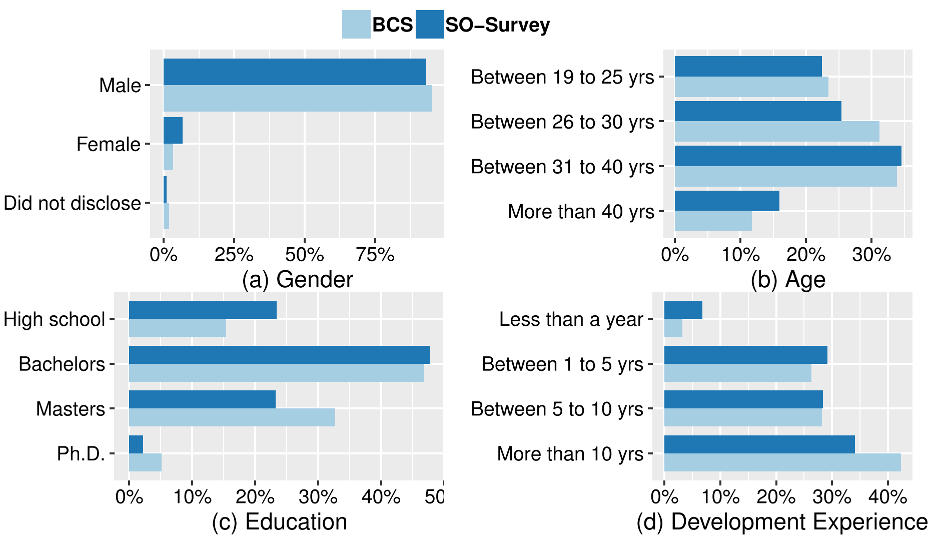


Fig. 3 Personal characteristics of the BCS developers

In terms of software development experiences (Fig. 3d), 70.5% of our respondents have more than five years of development experiences and 42.3% have more than 10 years. The corresponding numbers in the SO survey are 62.3% and 34.1% respectively. It indicates that the BCS developers are likely to be more experienced in software development than their non-BCS counterparts.

Key takeaway 1: In general, BCS developer population is more qualitifed than the general software developer population. Although, we noticed that more than 81% of our respondents have less than 2 years of BCS development experience, more than 70% developers from the same group were found to have more than five years of development experiences. These numbers indicate that a large number of software developers, who are experienced in non-BCS development, have recently joined BCS projects potentially due to the recent hypes generated by the blockchain technology.

6.2 RQ2: Motivations of BCS Developers

Figure 4 shows the primary motivations of BCS developers, which emerged from the answers to Q10 (Table 1) of our survey. Besides **technical attraction**, the other five categories of motivations that BCS developers have, are similar to the motivations of Open Source Software (OSS) developers (Hars and Ou 2001; Lakhani and Wolf 2003; Ye and Kishida 2003). Since blockchain is a new technology, many BCS developers indicate their fascinations to this innovative technology as one of their primary motives. While **ideology** did not top the list of OSS developers' motives (Hars and Ou 2001; Lakhani and Wolf 2003; Ye and Kishida 2003), it ranks top for the BCS developers.

Since results from the psychology domain suggest that a person's motivation may vary based on his/her age, education level (Hitka and Balážová 2015), compensation (Hennessey and Amabile 1998), or gender (Meece et al. 2006), we investigated whether those factors had any impact on a BCS developer's motives. Our results suggest significant differences (Chi Square, after applying False Discover Rate (FDR) corrections (Benjamini and Hochberg 1995) for multiple comparisons) in motivations based on the age ($\chi^2 = 33.34$, $p = 0.03$) or compensation ($\chi^2 = 40.58$, $p = 0.01$) of a respondent but no significant difference was found based on the level of education ($\chi^2 = 22.40$, $p = 0.29$) or gender ($\chi^2 = 9.83$, $p = 1.0$). While most of the BCS developers between 31 to 40 years of age have ideological

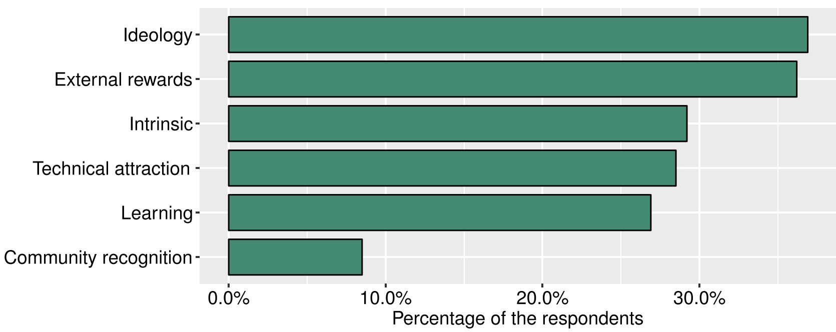


Fig. 4 Primary motivations of BCS developers'

motives, developers aged between 26 to 30 years are more likely to be motivated by external rewards and developers aged 25 years and younger are more likely to be motivated by the prospects of learning (Fig. 5). On the other hand, developers receiving some types of direct compensations are more likely motivated by external rewards. The following subsections examine these motivations in more detail.

6.2.1 Ideology

The primary motivation behind Bitcoin, the first blockchain based cryptocurrency was to create a decentralized currency that cannot be manipulated by a central authority. More than one-third of our respondents (36.9%) are motivated by a similar ideology.

I truly believe in the right to have a private way to send and store money. Also removing power from banks and governments. [#195]

6.2.2 External Rewards

Some developers (36.2%) contribute to BCS projects to earn money either by working part-time or by accepting bounty offers. Developers, who hold cryptocurrency, are naturally motivated to increase its value.

.. earn money; I get salary, and also I bought coins so their growth will give me money. [#147]

Many developers are full-time employees of the organization that manages his/her project.

I am paid to work full-time contributing to blockchain projects. [#75]

6.2.3 Intrinsic

Due to the various programming challenges that BCS development offers, many developers (29.2%), who enjoy writing code to solve problems, contribute to BCS projects.

I love coding. I love to solve problems and support people. [#127]

Some contribute to BCS projects due to their passions for a particular area.

It's an opportunity to work on a programming language, which I've always wanted to do. [#142]

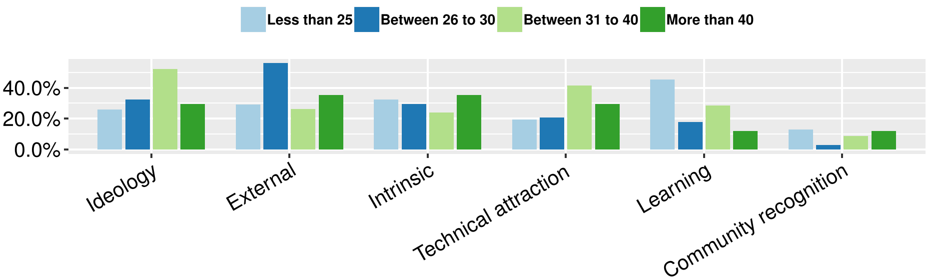


Fig. 5 Age vs. Motivation (percentage computed based on motivations within the same age group)

6.2.4 Technical Attraction

Attraction to the blockchain technology is one of the motivations for many BCS developers (28.5%).

> *Curiosity mainly. I was studying cryptography by myself before that and saw a chance to see it applied in unbelievable ways. [#31]*

6.2.5 Learning

Developers (26.9%) considering blockchain as a promising technology for the future want to learn and add it to their portfolio.

> *... to develop a better understanding of making and working of a blockchain. [#103]*

6.2.6 Community Recognition

Some respondents (8.5%) want to improve their portfolio through their involvement and recognition in the BCS community.

> *.. become more famous in the community (get good reputation). [#63]*

Key takeaway 2: Due to the significant financial gains by the early cryptocurrency investors as well as a large influx of cash through ICOs, we hypothesized that the majority of the BCS developers might be motivated by external rewards. However, the ratio of our respondents reporting external motives (36%) were similar to the number from prior OSS studies (Hars and Ou 2001; Lakhani and Wolf 2003). Moreover, the ratio of volunteers (43%) is also similar to what was reported in prior studies (Lakhani and Wolf 2003; Bosu et al. 2014a). On the other hand, ideological motives were more frequent among BCS developers (37%) than among the OSS developers (Hars and Ou 2001; Lakhani and Wolf 2003). Therefore, getting aligned with the *ideology* of a BCS community is important to become a member of that community.

6.3 RQ3: Differences Between BCS and Non-BCS Development

Our survey included two questions to find the differences between BCS and non-BCS development. Figure 6 shows the primary differences between BCS and non-BCS development as indicated by the respondents of our survey in response to Q11. Although BCS development has many similarities to traditional OSS development, due to some unique characteristics of the BCS domain, 93% of our respondents reported BCS development as somewhat different from non-BCS development.

Majority of our respondents did not encounter such strict security or reliability requirements while developing a non-BCS. The unique characteristics of the BCS domain (e.g., immutability, difficulty in upgrading the software) was also a differentiating factor for more than one-third respondents.

The following six subsections detail those differences. We conclude this section with a comparison of BCS development with three software development domains from Microsoft (Section 6.3.6).

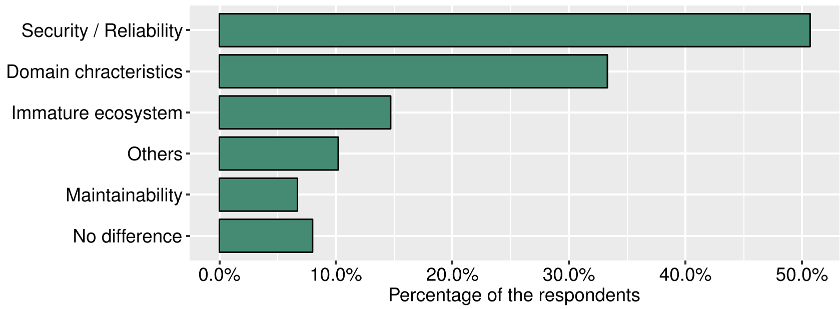


Fig. 6 Differences between BCS and non-BCS development

6.3.1 Security/Reliability

A higher emphasis on security and reliability is the primary factor that differentiates BCS from most of the non-BCS. Since the primary applications of the blockchain technology are maintaining ledgers of financial transactions, ensuring the security and reliability of a BCS application is the highest priority for majority of the BCS developers (50.7%).

Security and backward compatibility are held with utmost importance here unlike some other FOSS projects. [#3]

A single defect in a BCS application can cost millions of dollars. For example, recently a bug in the parity wallet code enabled hackers to steal $30M worth Ethereum tokens (Palladino 2017).

..you should be more careful because there is too much at stake (nowadays a lot of money are invested in cryptocurrencies). In most of the projects (non-blockchain) when a bug appears, it will be fixed and soon forgotten. But in blockchain projects some bugs can be very costly and never forgotten. [#147]

6.3.2 Domain Characteristics

One-third of our respondents (33.3%) consider several characteristics of the BCS domain as factors differentiating BCS development from non-BCS.

First, data stored in a blockchain is immutable. In other domains, there are several mechanisms to fix errors later by altering data. However, altering a blockchain ledger is almost impossible.

... also a disadvantage that you can not fix problems due to human errors or bugs in that transactions can not be changed. [#97]

Second, compared to the most non-BCS applications that operate on centralized and/or hosted environments, BCS applications operate on a complex, secured, distributed and decentralized network.

The distributed nature of blockchain software development makes it difficult to build robust software. Unreliable connections, unexpected latency, and malicious nodes create a hostile production environment. [#135]

Third, blockchains use public key cryptography and cryptographic hash functions to store and verify transactions. Cryptography is difficult to master and very few other domains require similar in-depth knowledge of cryptography as the BCS domain. Moreover, the knowledge of networking and networking security is a must for BCS development. The daunting requirement of having knowledge about diverse technological areas is another differentiating factor.

All kinds of knowledge are involved. It's hard to comprehend the whole project, including cryptography, network programming, economy policy etc.[#166]

Finally, the blockchain technology is changing rapidly with new protocols, innovations, and possibilities emerging everyday. BCS projects that are not able to evolve rapidly are at the risk of loosing their market capitalization.

Technology moves fast and blockchain software development (in general, not just Ethereum) is moving at an extremely fast pace. Need to keep up and adjust to meet whatever new requirements arise.[#98]

6.3.3 Immature Ecosystem

Many of the innovative aspects of the blockchain technology (e.g., smart-contract, privacy) are relatively new. Although the number of BCS projects have grown exponentially during the last couple of years, many tools and libraries that may support BCS development are still missing. Even the tools that currently exist are not stable. Therefore, many respondents (14.7%) consider the BCS development ecosystem as immatured compared to most of the non-BCS counterparts.

We are in 1986 on a web development timeline, to my mind. Like everything is in C can hit memory leaks, crashes, etc. Everything very raw and dangerous. We still have most of the stack to build. [#137]

Since Blockchain is a new technology, there is scarcity of enough domain-experienced developers compared to most non-BCS domains. The demographics of our survey also shows more than 83% respondents with less than two years of experience in BCS development.

People have more experience in other areas. [#48]

6.3.4 Maintainability

According to 6.7% of our respondents, maintaining a BCS application is difficult compared to a non-BCS application. Blockchains usually incorporate new functionality through hard forks (Wiki 2018a), which is usually scheduled way ahead of time to ensure that all the nodes are running the same version of the software. Due to the difficulty in upgrade, a BCS application needs to be well tested before a release.

.. non-blockchain software can easily upgrade features but blockchain software needs to wait for the preparation of nodes all over the world to upgrade functionality. [#22]

BCS applications run on public blockchains that have thousands of blocks created over the years. Therefore, a BCS application must be backward compatible and capable of validating earlier transactions or executing smart-contracts deployed through earlier versions.

Ethereum is a public chain. So we have to consider backward compatibility all the time. [#28]

6.3.5 No Difference

Some BCS developers (8%) do not find much difference between BCS and non-BCS development.

Not that different from other performance-sensitive, high reliability software. [#9]

6.3.6 How Does BCS Development Differ from Software Development at Microsoft?

The 'Rating distribution' column in Table 3 shows the distributions of the ratings by our respondents for 16 statements (The 'Statement' column in Table 3). The leftmost bar indicates 'Strongly disagree,' the middle bar indicates 'Neutral,' and the rightmost bar indicates 'Strongly agree.' We do not plot the ratings from the Microsoft developers since a prior publication reports those (Murphy-Hill et al. 2014).

Since the Shapiro-Wilk (Shapiro and Wilk 1965) test indicates that those ratings significantly differ from a normal distribution, we use non-parametric statistics. For each of the statements, we compare the responses from BCS developers with the responses of Microsoft developers from three domains (i.e., Games, Office, and Others). The three 'p' columns show statistical significance of the differences based on the Mann-Whitney U test after applying corrections for multiple comparisons using the FDR method (Benjamini and Hochberg 1995). The three 'r' columns show the effect sizes estimated using Rosenthal's formula (Rosenthal 1994). Rosenthal (1996) also recommends interpreting those effect sizes as: $r \geq 0.5 \Rightarrow$ *large*, $r \geq 0.3 \Rightarrow$ *medium*, and $r \geq 0.1 \Rightarrow$ *small*.

Posthoc, we grouped the 16 statements into following four categories based on the theme of each statement.

A. *Quality Assurance:* BCS developers agree more than both *Games* and *Office* developers that their software is well tested using unit tests (QA3). They also consider BCS more thoroughly automated tested (QA2) than both *Games* and *Other*. However, for manual testing (QA1), we did not find any significant difference. From the remaining two statements, BCS developers agree more than the *Games* developers on bug diagnosis difficulty (QA4) but disagree with them on the difficulty of writing unit tests (QA5). These results indicate BCS developers' higher emphasis on automated testing to ensure the quality of their software.
B. *Development process:* Among the four groups, BCS developers have the most flexible deadlines (DP1). Since the costs of defects are very high and patching a defect after release is very difficult, BCS developers are more flexible in postponing a release. On the other hand, among the four groups they are the least strict in following a process (DP4).
C. *Software maintenance:* Among the four groups, BCS developers are the most focused on paying technical debts (SM2), which further emphasizes the importance of long term maintainability of a BCS application. For the remaining two statements the differences were not significant.
D. *Ideology/morale:* In terms of ideology/morale, BCS has no significant difference with *Office*. The results of comparisons with the remaining two groups are mixed. On taking pride outside, for their work (DM4), BCS developers agree less than *Games* but are not

Table 3 Comparison of BCS development with three non-BCS domains at Microsoft

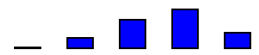
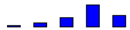
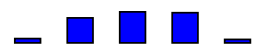
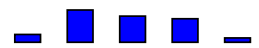
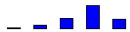
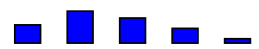
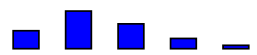
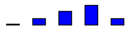
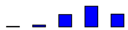
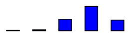
	Statement	Rating Distribution	BCS Vs. Games		BCS Vs. Office		BCS Vs. Other	
			p	r	p	r	p	r
A. Quality assurance								
QA1.	My software is well tested manually (e.g., paid testers thoroughly use the software).		0.82	0.06	0.052	0.14	0.09	0.12
QA2.	My software is well tested by automated testing (e.g., scripts that thoroughly use the software).		< 0.01	**0.32**	0.09	0.12	0.02	**0.18**
QA3.	My software is well tested by unit tests.		< 0.01	**0.34**	< 0.01	**0.31**	0.57	0.01
QA4.	Bugs in my software are hard to diagnose.		< 0.01	**0.23**	0.19	0.08	.34	0.06
QA5.	It's difficult to write thorough automated tests for my software because it's so complex.		0.02	**0.15**	0.13	0.1	0.93	0.1
B. Development process								
DP1.	My team has flexible release deadlines.		< 0.01	**0.46**	< 0.01	**0.53**	< 0.01	**0.44**
DP2.	My software has clear functional requirements.		0.02	**0.14**	0.64	0.02	0.93	0.09
DP3.	My team uses a waterfall process, rather than an agile process.		0.04	**0.12**	0.65	0.03	0.09	0.08
DP4.	My team adheres strictly to a process (for example, scrum or waterfall).		< 0.01	**0.2**	< 0.01	**0.37**	< 0.01	**0.29**

Table 3 (continued)

	Statement	Rating Distribution	BCS Vs. Games		BCS Vs. Office		BCS Vs. Other	
			p	*r*	*p*	*r*	*p*	*r*
C. Software maintenance								
SM1.	My software has high technical debt (for example, a lot of hacks).		0.3	0.04	0.13	0.1	0.53	0.02
SM2.	The technical debt is likely to be paid down in the future (for example, through refactoring).		< 0.01	**0.18**	< 0.01	**0.23**	0.02	**0.17**
SM3.	My software's architecture evolves significantly as the software gets more mature.		0.26	0.05	0.06	0.09	0.51	0.03
D. Ideology / morale								
DM1.	My software creates value for society.		< 0.01	**0.26**	0.21	0.07	0.02	**0.17**
DM2.	Creativity is highly valued on my team.		0.03	**0.14**	0.48	0.02	0.53	0.02
DM3.	Creating my software is challenging.		0.24	0.06	0.56	0	0.74	0.03
DM4.	When I tell people outside of my company about the software I work on, they are impressed.		0.04	**0.13**	0.13	0.1	0.09	0.11

Grey background indicates a statistically significant difference. For the effect sizes (*r*), a value in red indicates BCS developers agreeing less with that statement, while a value in green indicates more agreement from them. If an effect size is statistically significant, the background color intensity of the cell indicates strength of the effect size with darker shades indicating larger effect sizes. For example, negative effects are presented by: i) small, medium, and large and positive effects are presented by: i) small, medium, and large background color. Statistical significance (p) is reported after applying correction for multiple comparisons using False Discovery Rate (FDR) (Benjamini and Hochberg 1995) significantly different than *Other*; while on creating value for the society (DM1), BCS developers top both. BCS developers also agree less than the *Games* developer on the valuation of creativity (DM2) within their team.

In summary, BCS has the most differences with *Games* (i.e., 12), followed by *Other* (i.e., 5) and the least with *Office* (i.e., 4). BCS development has higher emphasis on security, reliability, and maintainability but lower emphasis on following a process or deadline than the other three domains. Apart from those differences, BCS may not be much different from traditional development domains, especially widely used software, such as *Office*.

Key takeaway 3: So, is BCS development really different? The answer to this question will depend on whom we ask. It's true that BCS development has a very high emphasis on security and reliability, but many of the existing software development domains (e.g., financial transaction, air traffic controller, and nuclear power plant management) have a similar emphasis on security and reliability. If a developer's non-BCS experience is in high assurance software (Section 6.3.5), then he/she might find little differences. However, over 93% of our respondents' non-BCS experiences significantly differed from their experiences in BCS (Fig. 6). Our survey received responses from ≈ 10% of the most active developers from the top BCS projects, and 70% of our respondents have more than 5 years of development experiences (Section 6.1). Yet, they found BCS development different from non-BCS. Some of the differences, such as the immaturity of the ecosystem, will resolve with time, but others, such as immutability of data as well as difficulty in upgrading the software after deployment, which is rare among the non-BCS domains, will linger as a differentiating factor.

6.4 RQ4: Challenges of BCS Development

Figure 7 shows the primary types of challenges of BCS development according to the respondents of our survey (Q13: Table 1). Since prior results from the Psychology domain suggest that a person's ability to acquire knowledge of a new domain depends on his/her age and education level (Beier and Ackerman 2005), we investigated whether those factors have any impact on the challenges reported by the BCS developers from our survey. Although we did not find any differences based on the education level ($\chi^2 = 14.46$, $p = 0.27$), we found that the developers aged 40 years and older, who are likely to have significant

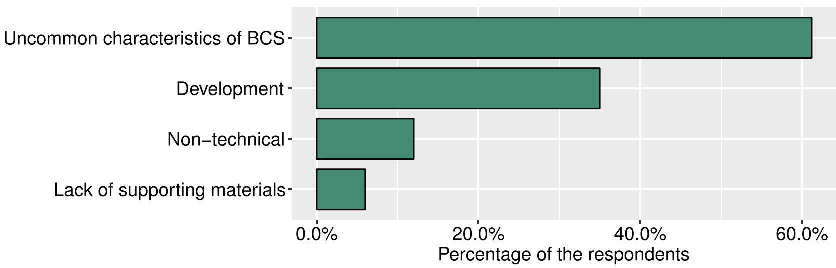


Fig. 7 Primary challenges of BCS development

experiences in non-BCS domains, were more likely ($\chi^2 = 35.07, p = 0.02$) to be challenged by the characteristics of the BCS domain (Section 6.4.1) than the other age groups.

6.4.1 Uncommon Characteristics of BCS

Some of the characteristics of the BCS domain that are rare among non-BCS, are sources of challenges for more than 61% BCS developers. Around 70% respondents of our survey have more than five years of software development experience. Therefore, the aspects of BCS development that differ from a non-BCS are primary sources of challenges for them. While discussing BCS development challenges, our respondents reiterated following differences already mentioned earlier (Section 6.3):

- high costs of defects (Section 6.3.1);
- technological complexity (Section 6.3.2);
- distributed, decentralized, and hostile environment (Section 6.3.2);
- rapidly changing ecosystem (Section 6.3.2); and
- difficulty in maintenance (Section 6.3.4).

Steep learning curves to get familiar with a BCS project is another challenge for many BCS developers. The codebase of a BCS project is not only complex but also requires a sound understanding of cryptography, networking, distributed systems as well as project specific protocols.

The bitcoin core codebase is an incredibly complicated system, so the hardest part is building up a good enough understanding of it such that it's safe to make changes. [#156]

6.4.2 Development

Development challenges are related to testing, ensuring security, and reviewing code as mentioned by 35% of our respondents.

Testing blockchain software is challenging as the software executes on a distributed and potentially hostile environment that currently cannot be adequately simulated on a development machine.

.. to test that bugs are not included in important parts such as consensus. [#22]

The security of a BCS software is the highest priority as it handles financial data that can be exploited for financial gains. Therefore, BCS developers must consider security aspects when writing code.

... thinking with a security mindset. the software is literally money, so when writing it you have to be wary of any way in which things could go wrong.[#143]

Due to the lack of quality reviewers, open source software (OSS) developers often have to wait for a long time to get their code reviewed (Bosu and Carver 2014b). BCS developers also report similar challenges.

... it is often difficult to get code reviews on open source projects even with a history of contributions. [#75]

6.4.3 Non-technical

The non-technical challenges of BCS development, which were reported by 12% of our respndents, are due to collaboration issues, difficulties in reaching agreement among the community members, and the ethical aspects of BCS.

Most of the BCS projects are run by communities. However, many of the projects have raised over hundreds of millions of dollars through initial coin offerrings (Howell et al. 2018) and have very high valuations (Cryptocurrency market capitalization 2019). Since a large sum of money is at stake, community members often engage in arguments to decide a project roadmap (De Filippi and Loveluck 2016).

Governance, deciding on changes, and set a future roadmap. [#35]

The development team comprising volunteers with diverse personalities, background, experience, and motivation often encounter issues to collaborate.

nothing particular to the blockchain, prickly personalities! [#51]

Some BCS developers are primarily motivated by the prospects of financial gains and will not hesitate to adopt an unethical approach if presented with an opportunity. Without a central monitoring authority, preventing developers with malicious intents is a challenge.

human greed, and cleaning up the mess after bounties and scams on forum.[#44]

6.4.4 Lack of Supporting Materials

The lack of supporting tools and documentation is a source of challenges for many BCS developers (6%).

Many of the supporting tools that can help their development tasks are yet to be developed. Moreover, the tools that are currently available are immature and unreliable.

Reliability of and the lack of good development tools. Like testing frameworks, and the difficulty of debugging.[#16]

To understand a complicated domain such as BCS, developers are looking for good learning materials and tutorials, but such materials are currently rarities. Even the documentation that they currently have are not user friendly.

Blockchain is totally a new technology so there is few information we can find. [#28]

Key takeaway 4: Since most of the non-BCS domains do not have similar high reliability and security requirements as the BCS domain, developers coming from other domains (except high assurance software) will encounter challenges due to those differences. Moreover, BCS Developers must be careful in writing code due to high costs of defects as well as difficulty in upgrading the software. Yet, they are under constant pressure due to a rapidly changing ecosystem as well as high expectations from the stakeholders to release new versions. Moreover, as the blockchains become more popular the scalability of BCS software has become an area of concern (Porru et al. 2017). As a result BCS development is challenging even for developers with considerable non-BCS experiences (Section 6.4.1).

6.5 RQ5: Tools that BCS Developers Need

We asked the respondent of our survey to describe the type of tools that they currently do not have but if implemented can significantly improve their development productivity (Q14: Table 1). In response, our respondents indicated their needs for four categories of tools (Fig. 8). We also hypothesized the the requirement for supporting tool may vary based on the software development experience or BCS development experience of a respondent. However, the results of our analysis did not find any statistically significant difference.

6.5.1 Testing Support

Majority of our respondents (53.1%) suggest that *testing* is the area of BCS development that currently needs supporting tools the most. BCS developers use ‘testnets’ (Wiki 2018b) to experiment their code before deploying it to the ‘mainnet’ (Documentation 2018). Many developers experience difficulties in setting up a testnet. An easy to setup, ‘one-click testnet’ may be a solution.

> *Easy way of forking mainnets for testing purposes, a way to deploy a test net in one click would be nice.[#150]*

The simulators that BCS developers currently have are limited and are unable to simulate a complex and hostile real world environment. While they have testnets, but the scales and complexities of testnets are no where closer to a mainnet. Few recent works (Stoykov et al. 2017; Chen et al. 2017) have attempted to build such simulators, but no satisfactory solution exists yet.

> *Easier ways to simulate complex network topologies on one single machine to simulate the network. [#195]*

Formal verification (Clarke and Wing 1996) techniques have been useful to secure BCS projects. However, developers find current formal specification languages (e.g., TLA+, VDM, and Z-notation) very complex to learn and use. They wish for user friendly alternatives.

> *End-to-end formal specification and verification tools with notations that mere programmers can understand. I sometimes think our formalists actually like to make their work obscure. [#1]*

Static analysis and penetration testing tools have been useful in non-BCS domains for security testing (Chess and McGraw 2004; Arkin et al. 2005). Since, those tools do not work well on BCS codebase, developers wish for automated security testing tools designed specifically for the BCS domain.

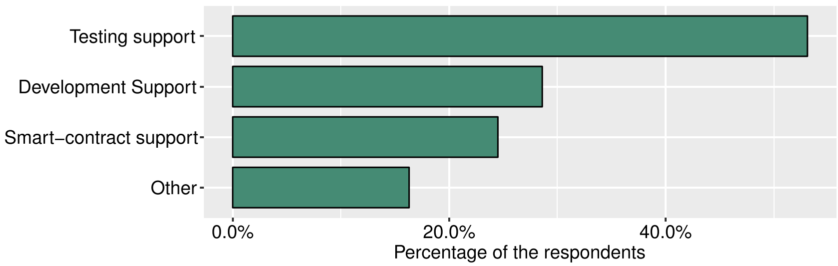


Fig. 8 Categories of tools that BCS developers need

Fuzz testing, something like linting for security best practices. [#196]

6.5.2 Development Support

The IDEs, designed for the non-BCS domains, lack adequate supports for testing and debugging a BCS codebase. Therefore, BCS developers use an array of tools for various development activities. Some developers (28.6%) maintain that an IDE designed specifically for BCS development would help them.

> *... they are all mostly disconnected, I have to switch from one tool/platform to another for my regular development activities many times every day. It would help if there were better integration between them all. [#66]*

Even the tools that exist are not reliable, and BCS developers wish them to be more stable.

> *Tools exist that are in their infancy and just need to get better. [#13]*

6.5.3 Smart-contract Development

Smart-contracts are written using contract-oriented programming language such as Solidity or Vyper and then compiled into bytecode for a platform (e.g., Ethereum Virual Machine aka EVM). Remix, an IDE for solidity, currently lacks many features such as: error highlighting and line by line debugging, that many BCS developers (24.5%) wish for.

> *Better tool support for smart-contract development. IDE integration with interactive debugging. [#137]*

Before interacting with a smart-contract, a developer might want to verify its security properties by decompiling its bytecode. While few solutions exist (Suiche 2017), smart-contract developers wish for a reliable and user-friendly decompiler.

> *... high-level Solidity decompiler that works (the current EVM-to-Solidity decompilers are horrible). [#113]*

6.5.4 Others

The other tools wished by the respondents include UML/design notations for the BCS domain, containers for deployment, and automated performance analysis tools.

Key takeaway 5: Based on the personal experiences of our respondents, they found some widely used tools tuned for non-BCS development, lacking required support for BCS development. While some of the needs expressed by our respondents (e.g., easy to write formal specification) may be a wishful thinking and difficult to achieve, most of those tools are feasible. Potentially implementable tools for BCS development include: testing environment, automatic security testing, static analysis for smart-contrats, and easy to deploy testnets. Since an array of research predict tremendous impacts of the blockchain technology and smartcontracts in future (Iansiti and Lakhani 2017; Peters and Panayi 2016; Fanning and Centers 2016), the number of BCS projects and developers contributing to those projects will grow significantly over the next decade. Therefore, research and development efforts should focus on implementing those tools to build a mature BCS development ecosystem.

7 Implications

In the following subsections, we discuss key implications of our results.

7.1 Notes for Prospective Joiners

Ideology drives more than one-third of the BCS developers (Section 6.2.1). Prior research suggest that OSS developers who do not share a common ideology with the community are not only less productive (Stewart and Gosain 2006) but also fail to form synergies with the existing members. There are several instances where ideological conflicts have split a BCS community into two rival projects (e.g., Ethereum → Ethereum and Ethereum Classic). Therefore, a prospective joiner should select a project that has ideologies aligned with his/her beliefs.

Second, an experienced non-BCS developer, regardless of his/her prior non-BCS domain, will encounter differences due to several unique characteristics of the BCS domain (Section 6.3.2). A prospective joiner must be prepared to conduct rigorous testing to ensure the correctness of code before deployment, since– i) data cannot be altered once written on the blockchain and ii) updating BCS is very difficult compared to non-BCS. Despite such extensive correctness requirements, tools and frameworks to conduct such rigorous tests are either limited or non-existent (Section 6.5.1). Therefore, a prospective joiner must be ready to spend significant manual efforts to test BCS for correctness, as our respondent#29 stated, *"Testing is 80% of development."*

Finally, BCS has steep learning curve with knowledge requirements in Cryptography, networking, security and distributed computing (Section 6.4.1). Yet, documentations and tutorials to assist the learning process of a newcomer are limited (Section 6.4.4). Therefore, a BCS joiner needs to be prepared to spend considerable efforts researching blockchain concepts and to utilize community Q&A sites (e.g., https://bitcoin.stackexchange.com).

7.2 Education and Preparation

Our results suggest that special skills, beyond those required for most non-BCS development would be beneficial for persons considering to join BCS. Chief among them is the knowledge of secured programming (Section 6.3.1). Unlike the majority of non-BCS, every module of a BCS is subject to attacks by malicious actors, since the stakes are substantially higher (Section 6.3.2). Second, due to its distributed and untrusted environment, a BCS developer needs in-depth networking knowledge to design and secure communication between distributed nodes (Section 6.3.2). Third, a sound understanding of cryptography is required, since BCS is secured using encryption (Section 6.3.2). Finally, a lot of concepts of Blockchain are verified using principles of mathematics, therefore a sound math skill is essential to design new BCS algorithms or protocols (Section 6.3.2).

7.3 Suggestions for tool Developers

The study outcome also presents a requirement for developing an array of tools specialized for BCS development. As discussed in Section 6.5, potentially implementable tools for BCS development include: testing environment, debugging tools, automatic security testing, static analysis for smart-contracts, and easy to deploy testnets.

Since several characteristics of the BCS domain are different from a non-BCS domain, quite a few categories of tools (e.g., IDE, simulator, and static analysis tools), that are stable and mature for non-BCS development, either do not exist or lacks important support for BCS (Section 6.5.1). Due to the lack of appropriate testing supports, BCS developers primarily depend on manual code reviews to ensure the security of their software (Chakraborty et al. 2018). As a result, expensive bugs and hacks are very frequent in the BCS ecosystem (Technologies 2017; Wan et al. 2017; Porru et al. 2017).

Smart-contracts, which open a new type of programming paradigm, are gaining popularities (Luu et al. 2016). Yet, the smart-contract programming domain lacks basic development supports, such as an IDE with error highlighting, line by line debugging, and decompilers (Section 6.5.3). As a result, many of the smart-contracts deployed on public blockchains are vulnerable (Nikolic et al. 2018).

Finally, due to a higher emphasis on security and reliability, BCS developers are highly interested in writing formal specifications for their software. However, current formal verification languages and tools are difficult to learn and use for BCS (Section 6.5.1). Therefore, another potential direction could be building customized formal specifications for the BCS domain.

7.4 Research Directions

The needs of BCS developers as identified in Section 6.5 will provide guidance to researchers to identify areas needing the most attentions. One of such areas is testing, where BCS development would benefit from research efforts. While prior research has focused on testing distributed programs (Campion and Messinger 2013), BCS operates on a different type of distributed environment where nodes are both collaborating and competing with each other other at the same. Moreover, some of those nodes may potentially have malicious intents. To simulate such networks in a testing environment (Section 6.5.1), research efforts are required to model their behavior. Such models may be also useful to mathematically verify the reliability and security of a network.

Second, smart-contracts automatically execute on distributed virtual-machines (e.g., EVM) based on certain predefined conditions. Since smart-contract programming languages (e.g., Solidity and Vyper) have different syntax and execution models, existing formal specification languages do not work well for those (Section 6.5.3). Although recent studies have focused on building new customized specifications to secure smart-contracts (Hildenbrandt et al. 2018; Park et al. 2018), more research effort is warranted as it is a high priority for the BCS developers.

Third, automated security testing is another key need for BCS developers. Fuzzing[4] technique has been successfully used for automated security testing in various domains. ContractFuzzer (Jiang et al. 2018), a recent tool, is the first fuzzer that aims to provide fuzzing support for Ethereum smart-contracts. However, such fuzzers are currently non-existent for other types of BCS. Research efforts to build fuzzers for various BCS would tremendously help the BCS ecosystem (Section 6.5.1) as security is the top priority for BCS projects. Finally, academic researchers, often in collaboration with practitioners, may focus on automated documentation generation, the lack of which is a barrier for the newcomers (Section 6.4.4).

[4] Providing invalid, unexpected, or random data as inputs.

8 Threats to Validity

This section discusses the four common types threats to validity for this study.

8.1 Internal Validity

Participant selection is the primary threat to internal validity. We selected five commits as the threshold for an invitation to this survey. A higher or lower threshold may have altered the results. However, the results of our analysis did not find any significantly differences in opinions based on the number of pull requests of a respondent (Q8). Therefore, this threat may be minimal.

Although the response rate of our survey is similar to prior SE surveys (Bosu et al. 2017; Kononenko et al. 2016), the response rate is only (≈13%). Therefore our results could suffer from a potential ‘non-response bias’ (i.e., the opinions of the respondents who chose to participate may be different from who did not) (Armstrong and Overton 1977). Even so, the 156 responses that we analyzed provide a rich source of data to reveal the insights described in this paper.

8.2 Construct Validity

The primary threats to construct validity are related to the design of our survey. For example, respondents may misunderstand our questions or the questions may not be appropriate to investigate our research questions. Therefore, we took following measures during our survey design to reduce potential threats.

- We conducted expert reviews as well as a pilot test to check both understandability and appropriateness of the survey questions.
- We carefully worded the open-ended questions in an unbiased manner.
- We provided clear instructions as well as asked the respondents to answer based on their personal experiences with BCS projects.

The actual responses our open-ended questions do indicate that the respondents understood the intent of the survey questions. Therefore, we think this threat is minimal.

Another potential threat to construct validity is our comparison of the BCS developers’ ratings collected for this study with the ratings from the MS study. On the comparability of two different surveys, Herndon suggests that two surveys are comparable if both surveys have: i) the same target population, ii) the same objective, and iii) uses the same survey methodology (Herndon 2018). The MS study, empirically identified 28 statements spanning various software engineering as well as general work feature, to identify differences between game and non-game development. These statements were rated by three separate groups of developers to identify differences in opinions among those groups. In this study, we have collected ratings from a fourth group (i.e. BCS developers) for 19 out of the 28 statements that are applicable to BSC projects and compared that group with the three groups from the MS study. We believe data collected for our study is comparable with the MS study since both studies have: i) software developers as the target population, ii) have the same objective of comparing the opinions of software developers from different domains, and iii) used a web-based survey to solicit ratings for a similar set of statements using the same Likert-scale. Moreover, comparing data from multiple surveys is not uncommon in many disciplines (Elliott and Davis 2005; Sadana et al. 2002; Schenker et al. 2016; Lohr and Rao 2000; Raghunathan et al. 2007). Therefore, we consider this threat to be minimal.

Finally, we have compared the demographics of our respondents against the demographics of the respondents reported in the SO survey. Comparisons of demographics across multiple surveys is common (Mcdonald 2007; Ross et al. 2010; Posel and Devey 2006). Since our respondents come from the same population (i.e., software developers), a possible threat to validity for these comparisons is minimal.

8.3 External Validity

The respondents of our survey may not adequately represent all BCS developers. While our respondents come from 61 different BCS projects, they primarily represent the top ones. Therefore, some of the opinions, especially, the challenges encountered by BCS developers in smaller projects may be different from those included in this study. However, our results also indicate that BCS developers' challenges are primarily due to the differences between BCS and non-BCS development, which may be similar across all projects.

Section 6.3.6 compares BCS development with three non-BCS development domains from the same organization (i.e., Microsoft). Therefore, differences identified in that section may not apply to a different domain or a different organization. A common misconception about industrial research at large companies such as Microsoft is that software projects at Microsoft are not representative of other software projects. While projects might be larger in size, most development practices at Microsoft are adapted from the general software engineering community and also used outside Microsoft. Since Microsoft is a well-recognized and mature software development organization, we believe that MS developers are eligible to provide us credible insights on non-BCS development,

8.4 Conclusion Validity

We sent out the survey to 1,604 BCS developers that were eligible at that time. Using Yamane's (1973) formula for a recommended sample size, we would need ≈320 responses to obtain results that are within 5% margin of error. Despite our best efforts, we obtained only 156 responses that are eligible for analyses. Therefore, many of our quantitative results are subject higher margin of errors. However, several prior SE surveys that have provided valuable insights to the community are also subject to similar errors as responses for SE surveys are generally low (Lee et al. 2017; Kononenko et al. 2016; Bosu et al. 2017).

9 Conclusion

Despite the existence of a large number of active BCS projects as well as tremendous developer interests in the blockchain technology, there have been few empirical SE research exploring this area. In an attempt to bridge this gap, we studied the motivations, challenges, and needs of BCS developers. Our results suggest that although most of the BCS projects are Open Source Software (OSS) projects by nature, more than 93% of our respondents found BCS development somewhat different from a non-BCS development as BCS projects have higher emphasis on security and reliability than most of the non-BCS projects. Other differences include: higher costs of defects, decentralized and hostile environment, technological complexity, and difficulty in upgrading the software after release. These differences were also the primary sources of challenges to the BCS developers. Software development tools that are tuned for non-BCS do not adequately support BCS development tasks and the ecosystem needs an array of new or improved tools, such as: a customized IDE for

BCS development tasks, debuggers for smart-contracts, testing support, easily deployable simulators, and BCS domain specific design notations.

From the findings of our study, the prospective joiners of BCS development may gain some technical insight such as to learn secure programming practices, form mindset on test-centric development approach and also non-technical issues related to collaboration in the community of diverse motivation. Students wishing to join BCS should prepare themselves by gaining knowledge in networking, distributed programming, cryptography, and mathematics. This study also identifies pressing need of tools supporting various BCS development tasks. Many of these tools require active participation of SE researchers who are expected to grab high-impact research problems from this research.

Acknowledgements We are grateful to Emerson Murphy-Hill from Google; Tom Zimmermann, and Nachi Nagappan from Microsoft Research for providing us the dataset of their Survey at Microsoft. We thank the anonymous reviewers for their thorough reviews and highly appreciate the comments and suggestions, which significantly contributed to improving the quality of this publication. We thank the anonymous respondents of our survey.

Appendix

Table 4 Codes that emerged from our open-coding of the four survey questions and the categories that we assigned each code to

Question	Codes	Assigned category
Q10: What are your motivations to contribute to your primary project?	Ideology	Ideology
	Financial gains	External rewards
	Job, profession	
	Hobby, Fun	Intrinsic motivation
	Passion, Self interest	
	Technical attraction	Technical attraction
	Learning, professional development	Learning
	Community recognition	Community Recognition
Q11: Based on your experiences, what are the primary differences between blockchain and non-blockchain software development ?	High emphasis on security reliability	Security/ reliability
	Irreversible data	
	Costly defects	
	Networking knowledge	Domain characteristics
	Cryptography knowledge	
	Distributed environment	
	High pace development	
	Lack of tools	Immature ecosystem
	New technology /framework	
	Lack of experienced developer	
	Backward compatibility	Maintainability
	Upgrade difficulty	

Table 4 (continued)

Question	Codes	Assigned category
	No difference	No difference
	Others	Others
Q13: What are the most challenging aspects of blockchain software development?	Cost of defects	Uncommon characteristics of BCS
	Steep learning curve	
	Technological complexity	
	Maintainability	
	High pace development	
	Scalability	
	Testing	Development challenges
	Security	
	Code reviews	
	Collaboration difficulties	Non-technical challenges
	Governance, politics	
	Ethical aspects	
	Lack of tools	Lack of supporting materials
	Lack of documentation	
Q14: Please describe the type of tools that you currently do not have, but if implemented, can greatly help your blockchain software development activities.	Simulator	Testing supports
	Security testing support	
	Easily deployable testnet	
	Formal spec verifier	
	Static analysis tools	
	Development environment /IDE	Development supports
	Stable development tools	
	Easy to write formal specification	
	IDE for smart contracts	Smart-contract support
	Debugger for smart contracts	
	EVM decompiler	
	Blockchain specific design notations	Others
	Container	
	Others	

References

Abran A, Moore JW, Bourque P, Dupuis R, Tripp L (2004) Software engineering body of knowledge. IEEE Computer Society, Angela Burgess

Adhami S, Giudici G, Martinazzi S (2018) Why do businesses go crypto? An empirical analysis of initial coin offerings. Journal of Economics and Business

Arkin B, Stender S, McGraw G (2005) Software penetration testing. IEEE Secur Priv 3(1):84–87

Armstrong JS, Overton TS (1977) Estimating Nonresponse Bias in Mail Surveys. J Market Res 14(3):396–402
Azaria A, Ekblaw A, Vieira T, Lippman A (2016) Medrec: Using blockchain for medical data access and permission management. In: International Conference on Open and Big Data (OBD). IEEE, pp 25–30
Beier ME, Ackerman PL (2005) Age, ability, and the role of prior knowledge on the acquisition of new domain knowledge: promising results in a real-world learning environment. Psychol Aging 20(2):341
Benjamini Y, Hochberg Y (1995) Controlling the false discovery rate: a practical and powerful approach to multiple testing. Journal of the royal statistical society. Series B (Methodological) 57(1):289–300
Bosu A, Carver J, Guadagno R, Bassett B, McCallum D, Hochstein L (2014a) Peer impressions in open source organizations: a survey. J Syst Softw 94(0):4–15
Bosu A, Carver JC (2014b) Impact of developer reputation on code review outcomes in oss projects: an empirical investigation. In: Proceedings of the 8th ACM/IEEE international symposium on empirical software engineering and measurement. ACM, pp 33
Bosu A, Carver JC, Bird C, Orbeck J, Chockley C (2017) Process aspects and social dynamics of contemporary code review: insights from open source development and industrial practice at microsoft. IEEE Trans Softw Eng 43(1):56–75
Brooke S (2018) The ins and outs of testing blockchain apps. https://jaxenter.com/ins-outs-testing-blockchain-apps-146447.html
Campion S, Messinger D (2013) System and method for distributed software testing. US Patent 8,621,434
Chakraborty P, Shahriyar R, Iqbal A, Bosu A (2018) Understanding the software development practices of blockchain projects: a survey. In: Proceedings of the 12th ACM/IEEE international symposium on empirical software engineering and measurement, ESEM ’18, pp 28:1–28:10
Chen C, Qi Z, Liu Y, Lei K (2017) Using virtualization for blockchain testing. In: International conference on smart computing and communication. Springer, pp 289–299
Chess B, McGraw G (2004) Static analysis for security. IEEE Secur Priv 2(6):76–79
Chuen DLK (2015) Handbook of digital currency: bitcoin, innovation, financial instruments, and big data. Academic Press, New York
Clack CD, Bakshi VA, Braine L (2016) Smart contract templates: foundations, design landscape and research directions. arXiv:1608.00771
Clarke EM, Wing JM (1996) Formal methods: State of the art and future directions. ACM Comput Surv (CSUR) 28(4):626–643
Cohen J (1960) A coefficient of agreement for nominal scales. Educ Psychol Meas 20(1):37–46
Cryptocurrency market capitalization (2019) https://www.coinmarketcap.com/. Accessed: 2019-01-15
Dahlander L, Mckelvey M (2005) Who is not developing open source software? non-users, users, and developers. Econ Innov Technol 14(7):617–635
David PA, Shapiro JS (2008) Community-based production of open-source software: What do we know about the developers who participate? Inf Econ Policy 20(4):364–398. Empirical Issues in Open Source Software
De Filippi P, Loveluck B (2016) The invisible politics of bitcoin: Governance crisis of a decentralized infrastructure. Internet Policy Rev 5(4):529–546
Deci E, Ryan RM (1985) Intrinsic motivation and self-determination in human behavior. Springer Science & Business Media, Berlin
Delmolino K, Arnett M, Kosba A, Miller A, Shi E (2016) Step by step towards creating a safe smart contract: Lessons and insights from a cryptocurrency lab. In: International conference on financial cryptography and data security. Springer, pp 79–94
Destefanis G, Marchesi M, Ortu M, Tonelli R, Bracciali A, Hierons RM (2018) Smart contracts vulnerabilities: a call for blockchain software engineering?. In: Proceedings of the 2018 international workshop on blockchain oriented software engineering. IEEE, Campobasso, pp 19–25
Documentation BD (2018) Mainnet, bicoin main network. https://bitcoin.org/en/glossary/mainnet. Accessed: 2018-01-04
Elliott MR, Davis WW (2005) Obtaining cancer risk factor prevalence estimates in small areas: combining data from two surveys. J Royal Stat Soc: Ser C (Appl Stat) 54(3):595–609
Fanning K, Centers DP (2016) Blockchain and its coming impact on financial services. J Corp Account Financ 27(5):53–57
Galati F (2019) Blockchain as a process: Ideologies and motivations behind the technology. https://medium.com/coinmonks/blockchain-as-a-process-ideologies-and-motivations-behind-the-technology-c25219d87881/
Garfinkel SL (1996) Public key cryptography. Computer 29(6):101–104
Gilbertson T, Vroegindewey L (2017) Larry’s cryptocoin risk. https://www.coindevelopmentindex.com/

Hars A, Ou S (2001) Working for free? motivations of participating in open source projects. In: 2001. Proceedings of the 34th annual Hawaii international conference on system sciences. IEEE, pp 9–pp
Hennessey BA, Amabile TM (1998) Reality, intrinsic motivation, and creativity. Am Psychol 51:1153–1166
Herndon JB (2018) Comparing and linking survey data: Considerations for working with multiple data sources. https://www.nidcr.nih.gov/grants-funding/grant-programs/behavioral-social-sciences-research-program/comparing-and-linking-survey-data
Hertel G, Niedner S, Herrmann S (2003) Motivation of software developers in open source projects: an internet-based survey of contributors to the linux kernel. Res Policy 32(7):1159–1177
Hildenbrandt E, Saxena M, Rodrigues N, Zhu X, Daian P, Guth D, Moore B, Park D, Zhang Y, Stefanescu A et al (2018) Kevm: a complete formal semantics of the ethereum virtual machine. In: 2018 IEEE 31st computer security foundations symposium (CSF). IEEE, pp 204–217
Hitka M, Balážová Ž (2015) The impact of age, education and seniority on motivation of employees. Bus: Theory Pract 16:113
Howell ST, Niessner M, Yermack D (2018) Initial coin offerings: Financing growth with cryptocurrency token sales. Technical report, National Bureau of Economic Research
Huh S, Cho S, Kim S (2017) Managing iot devices using blockchain platform. In: 2017 19th International Conference on Advanced Communication Technology (ICACT). IEEE, pp 464–467
Humphrey SE, Nahrgang JD, Morgeson FP (2007) Integrating motivational, social, and contextual work design features: a meta-analytic summary and theoretical extension of the work design literature. J Appl Psychol 92(5):1332
Iansiti M, Lakhani KR (2017) The truth about blockchain. Harv Bus Rev 95(1):118–127
IBM Corp. Released (2017) IBM SPSS Statistics for Windows, Version 25.0. Armonk, NY: IBM Corp.
Jacobovitz O (2016) Blockchain for identity management
Jakobsson M, Juels A (1999) Proofs of work and bread pudding protocols. In: Secure information networks. Springer, pp 258–272
Jiang B, Liu Y, Chan W (2018) Contractfuzzer: Fuzzing smart contracts for vulnerability detection. In: Proceedings of the 33rd ACM/IEEE international conference on automated software engineering. ACM, pp 259–269
Kononenko O, Baysal O, Godfrey MW (2016) Code review quality: how developers see it. In: 2016 IEEE/ACM 38th international conference on software engineering (ICSE). IEEE, pp 1028–1038
Korpela K, Hallikas J, Dahlberg T (2017) Digital supply chain transformation toward blockchain integration. In: Proceedings of the 50th Hawaii international conference on system sciences
Krafft PM, Della Penna N, Pentland AS (2018) An experimental study of cryptocurrency market dynamics. In: Proceedings of the 2018 CHI conference on human factors in computing systems, CHI '18, pp 605:1–605:13
Lakhani KR, Wolf RG (2003) Why hackers do what they do: Understanding motivation and effort in free/open source software projects. MIT Sloan Working Paper 4425–03
Landis JR, Koch GG (1977) The measurement of observer agreement for categorical data. Biometrics 33(1):159–174. http://www.jstor.org/stable/2529310
Lee A, Carver JC, Bosu A (2017) Understanding the impressions, motivations, and barriers of one time code contributors to floss projects: a survey. In: Proceedings of the 39th international conference on software engineering, ICSE '17. IEEE Press, Piscataway, pp 187–197. https://doi.org/10.1109/ICSE.2017.25
Lohr SL, Rao J (2000) Inference from dual frame surveys. J Am Stat Assoc 95(449):271–280
Lu CJ, Shulman SW (2008) Rigor and flexibility in computer-based qualitative research: Introducing the coding analysis toolkit. Int J Multiple Res Approaches 2(1):105–117
Luu L, Chu DH, Olickel H, Saxena P, Hobor A (2016) Making smart contracts smarter. In: Proceedings of the 2016 ACM SIGSAC conference on computer and communications security. ACM, pp 254–269
Mcdonald MP (2007) The true electorate: a Cross-Validation of voter registration files and election survey demographics. Publ Opin Q 71(4):588–602
Meece JL, Glienke BB, Burg S (2006) Gender and motivation. J Sch Psychol 44(5):351–373
Mockus A, Fielding RT, Herbsleb JD (2002) Two case studies of open source software development: Apache and mozilla. ACM Trans Softw Eng Methodol 11(3):309–346
Murphy-Hill E, Zimmermann T, Nagappan N (2014) Cowboys, ankle sprains, and keepers of quality: How is video game development different from software development?. In: Proceedings of the 36th international conference on software engineering. ACM, pp 1–11
Nakamoto S (2008) Bitcoin: A peer-to-peer electronic cash system
Narayanan A, Bonneau J, Felten E, Miller A, Goldfeder S (2016) Bitcoin and cryptocurrency technologies: a comprehensive introduction. Princeton University Press, Princeton
Nikolic I, Kolluri A, Sergey I, Saxena P, Hobor A (2018) Finding the greedy, prodigal, and suicidal contracts at scale. arXiv:1802.06038

Overflow S (2017) Stack overflow annual developer survey. https://insights.stackoverflow.com/survey/
Palladino S (2017) The parity wallet hack explained. https://blog.zeppelin.solutions/on-the-parity-wallet-multisig-hack-405a8c12e8f7
Park D, Zhang Y, Saxena M, Daian P, Roşu G. (2018) A formal verification tool for ethereum vm bytecode. In: Proceedings of the 2018 26th ACM joint meeting on european software engineering conference and symposium on the foundations of software engineering. ACM, pp 912–915
Peters GW, Panayi E (2016) Understanding modern banking ledgers through blockchain technologies: Future of transaction processing and smart contracts on the internet of money. In: Banking beyond banks and money. Springer, pp 239–278
Porru S, Pinna A, Marchesi M, Tonelli R (2017) Blockchain-oriented software engineering: challenges and new directions. In: Proceedings of the 39th international conference on software engineering companion. IEEE Press, Buenos Aires, pp 169–171
Posel D, Devey R (2006) The demographics of fathers in south africa: an analysis of survey data, 1993–2002 Baba: men and fatherhood in South Africa, pp 38–52
Raghunathan TE, Xie D, Schenker N, Parsons VL, Davis WW, Dodd KW, Feuer EJ (2007) Combining information from two surveys to estimate county-level prevalence rates of cancer risk factors and screening. J Am Stat Assoc 102(478):474–486
Reijers W, Coeckelbergh M (2018) The blockchain as a narrative technology: Investigating the social ontology and normative configurations of cryptocurrencies. Phil Technol 31(1):103–130
Roberts JA, Hann IH, Slaughter SA (2006) Understanding the motivations, participation, and performance of open source software developers: a longitudinal study of the apache projects. Manag Sci 52(7):984–999
Rosenthal JA (1996) Qualitative descriptors of strength of association and effect size. J Soc Serv Res 21(4):37–59
Rosenthal R (1994) Parametric measures of effect size. In: Cooper H, Hedges LV (eds) The Handbook of Research Synthesis. Russell Sage Foundation, New York, pp 231–244
Ross J, Irani L, Silberman MS, Zaldivar A, Tomlinson B (2010) Who are the crowdworkers?: Shifting demographics in mechanical turk. In: CHI '10 Extended abstracts on human factors in computing systems, CHI EA '10, pp 2863–2872
Sadana R, Mathers CD, Lopez AD, Murray CJ, Iburg K (2002) Comparative analyses of more than 50 household surveys on health status. In: Lopez AD (ed) Summary Measures of Population Health: Concepts, Ethics, Measurement and Applications. World Health Organization, Geneva, pp 369–386
Schenker N, Gentleman JF, Rose D, Hing E, Shimizu IM (2016) Combining estimates from complementary surveys: a case study using prevalence estimates from national health surveys of households and nursing homes. Public Health Reports
Shapiro SS, Wilk MB (1965) An analysis of variance test for normality (complete samples). Biometrika 52(3/4):591–611
Simmons GJ (1979) Symmetric and asymmetric encryption. ACM Comput Surv (CSUR) 11(4):305–330
Snow J, Mann M (2013) Qualtrics survey software: handbook for research professionals. Qualtrics Labs, Inc
Stallman R (2003) Free software foundation (fsf)
Stewart KJ, Gosain S (2006) The impact of ideology on effectiveness in open source software development teams. Mis Quarterly 30(2):291–314
Stoykov L, Zhang K, Jacobsen HA (2017) Vibes: fast blockchain simulations for large-scale peer-to-peer networks. In: Proceedings of the 18th ACM/IFIP/USENIX Middleware Conference: Posters and Demos. ACM, pp 19–20
Suiche M (2017) Porosity: a decompiler for blockchain-based smart contracts bytecode. DEF CON **25**
Swan M (2015) Blockchain: Blueprint for a new economy. "O'Reilly Media Inc."
Technologies P (2017) A postmortem on the parity multi-sig library self-destruct. https://paritytech.io/a-postmortem-on-the-parity-multi-sig-library-self-destruct/
Underwood S (2016) Blockchain beyond bitcoin. Commun ACM 59(11):15–17
Von Krogh G, Haefliger S, Spaeth S, Wallin MW (2012) Carrots and rainbows: Motivation and social practice in open source software development. MIS quarterly 36(2):649–676
Wan Z, Lo D, Xia X, Cai L (2017) Bug characteristics in blockchain systems: a large-scale empirical study. In: 2017 IEEE/ACM 14th international conference on Mining software repositories (MSR). IEEE, pp 413–424
West J, Gallagher S (2006) Challenges of open innovation: the paradox of firm investment in open-source software. R & D Manag 36(3):319–331
Wiki B (2018a) Hardfork. https://bitcoin.org/en/glossary/hard-fork. Accessed: 2018-01-04
Wiki B (2018b) Testnet. https://en.bitcoin.it/wiki/Testnet. Accessed: 2018-01-04
Yamane T (1973) Statistics: An introductory analysis

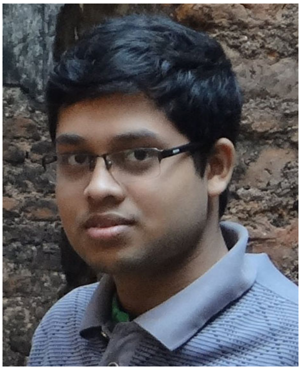
Ye Y, Kishida K (2003) Toward an understanding of the motivation open source software developers. In: Proceedings of the 25th international conference on software engineering. IEEE Computer Society, pp 419–429

Zheng Z, Xie S, Dai HN, Wang H (2016) Blockchain challenges and opportunities: A survey. Work Pap.–2016

Amiangshu Bosu is an Assistant Professor of Computer Science at the Wayne State University. He held a similar position at the Southern Illinois University Carbondale from 2016 to 2018. He completed his Ph.D. dissertation work at the University of Alabama in 2015 under the supervision of Dr. Jeffrey Carver and conducted postdoctoral research under the guidance of Dr. Danfeng Yao at the Virginia Tech. His research spans empirical software engineering, peer code review, software security, android security, malware detection, mining software repositories, and social network analysis. His research appears in prestigious conferences and journals in the areas of software engineering such as ICSE, FSE, EMSE, TSE, JSS, ASE, ESEM, SANER, and MSR. He is a professional member of the ACM and the IEEE.

Anindya Iqbal received B.Sc.Eng. (hons.) and M.Sc.Eng degrees in Computer Science and Engineering from Bangladesh University of Engineering and Technology (BUET), Bangladesh, in 2005 and 2009, respectively. He has received PhD degree from Monash University, Australia in 2013. He is an Associate Professor in the department of Computer Science and Engineering, Bangladesh University of Engineering and Technology (BUET). His major research interests are in the fields of Software Engineering, empirical software engineering, applied machine learning, participatory sensing system, security & privacy, and wireless sensor networks. He has published more than 20 refereed publications in reputed journals and conferences including Ad Hoc Networks, Journal of Network and Computer Application, ASE, ESEM, APSEC, WoWMoM, LCN, NCA, Trustcom, and WCNC.

Rifat Shahriyar is working as an Associate Professor at the Department of Computer Science & Engineering (CSE) of Bangladesh University of Engineering & Technology (BUET). He completed Doctor of Philosophy (Ph.D.) in April 2015 from Research School of Computer Science at Australian National University (ANU). He completed Bachelor of Science (B.Sc.) in May 2007 and Masters of Science (M.Sc.) in December 2009 from CSE, BUET. His research interests are memory management especially garbage collection, programming language, high-performance computing, and software engineering. He has published more than 15 refereed publications in reputed journals and conferences including ACM SIGPLAN Notices, OOPSLA, ISMM, ESEM, HPCC, and ICDCN.



Partha Chakraborty received BSc. Eng. degree in Computer Science and Engineering from Bangladesh University of Engineering and Technology (BUET) in 2018. His research interests are in mining software repositories, empirical software engineering, and machine learning.

Affiliations

Amiangshu Bosu[1] · Anindya Iqbal[2] · Rifat Shahriyar[2] · Partha Chakraborty[2]

Anindya Iqbal
anindya@cse.buet.ac.bd

Rifat Shahriyar
rifat@cse.buet.ac.bd

Partha Chakroborty
shuvopartho@gmail.com

1 Department of Computer Science, Wayne State University, Detroit, MI, USA

2 Department of Computer Science and Engineering, Bangladesh Uniersity of Engineering and Technology, Dhaka, Bangladesh