THINK INDIA JOURNAL

ISSN:0971-1260

Vol-22-Issue-41-December-2019

Securing the Software Development Life Cycle (SDLC) with a Blockchain Oriented Development Approach

Siddharth Lilani, Dhrumil Malani, Jimit Modi and Fenil Soni

B. Tech. Students, Department of Information Technology, SVKM's NMIMS Shirpur

ABSTRACT

The need of the hour for all software development companies today is to ensure a secure development environment for effective progress of work. The agile process model has been subject to changes by organizations which set standards of development for themselves and even other companies. This is done to ascertain the high security and privacy requirements of projects. This is also done to fortify better development ideas in the field of secure Agile development and to release new updates of the process model as open source ideas to interested parties.

Improving already existing process models helps in increasing the business concerns and risks associated with projects. Security related activities were only a part of the testing phase of the SDLC. Due to this, many issues used to be resolved at a very late stage and developers had to work on them from the point where development went wrong or from scratch altogether. Integration of activities across the SDLC to help discover and reduce vulnerabilities early is a better practice. This can be easily implemented by listing down security requirements alongside the functional requirements while performing risk analysis of the system.

This paper acknowledges the need for the development of specialized techniques for block-chain oriented Software development and testing and throws light on the issues which are faced in this process. The aim of this paper is to devise a way to put the SDLC on the blockchain and ensure its efficient testing. It also highlights the problems that developers face today, and the approaches followed for testing software to ensure the best delivery.

Index Terms: Blockchain, Blockchain-Oriented Software, SDLC, Software, Software Testing.

INTRODUCTION

BLOCKCHAIN technology has indeed revolutionized the way people though of Internet and has played a major role in bring about development in transactions online. A major reason it is being used widely is its security. It involves no external parties thereby creating a lot of trust. A blockchain is a ledger which is shared, replicated and synchronized among the members of a network. It is not centralized in nature. Any updates to the ledger are also stored chronologically with details of who made those changes and when and how they were made. All updates are time stamped and have a unique cryptographic signature to add authenticity to it. Tampering information in a blockchain isn't possible. The most recent block gets the most recently updated or generated information.

The decentralized nature of the blockchain has helped it gain a lot of attention of the masses. It allows the creation of decentralized teams consisting of developers, managers, business analysts, testers and dev-ops engineers who can together build software products without being at a single physical location. Considering traditional Software Development methodologies, the testing of a software being developed on the blockchain would be extremely complex. An efficient yet quick testing technique is essential because fixing bugs and any other major design issues in the software is what decides the performance of any firm which develops software.

Software engineers have an increased interest in blockchain development because of the popularity it gained so quickly. Various software implementations have been used over time to give birth to rapidly developing software. A lot many of these projects couldn't sustain for too long because they didn't keep in mind the basic principles of software engineering. It is necessary to put new ideas and methods to use but all basic concepts of software engineering must be kept in mind too.

ISSN:0971-1260

Vol-22-Issue-41-December-2019

LITERATURE REVIEW

Blockchain, if explained in simple words, can be described as a data structure in which information is stored with validation particulars for authentication. Data integrity and uniqueness are ensured to maintain trust as Blockchain-Oriented Software (BOS) is used for systems that need to handle critical security.

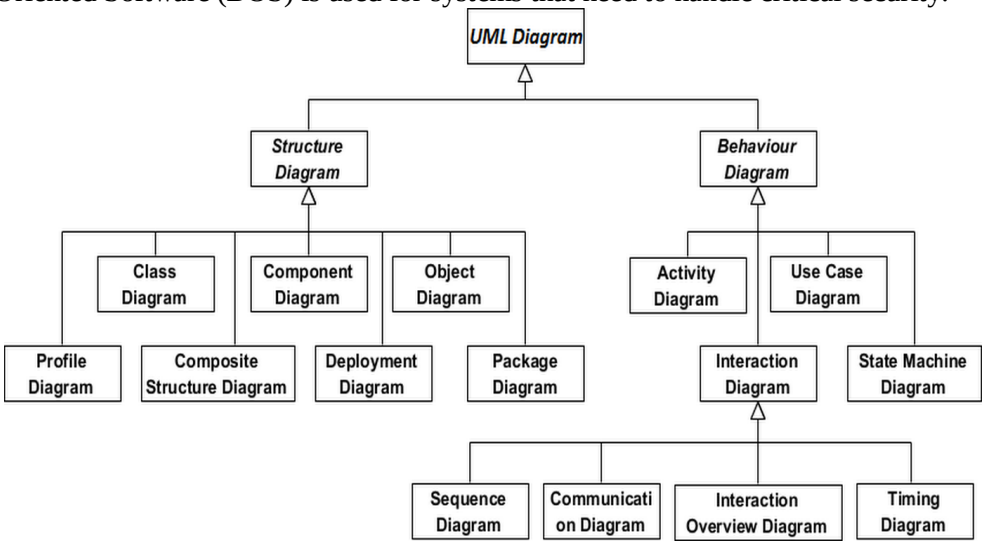

Figure 1: Composition of a UML Diagram

It is widely discussed that blockchain development has certain issues. Some experts, comparing it to regular mobile and web development, have even called it a disruptive computing paradigm thereby questioning it critically. Mobile phone specific software development has gained popularity in the recent years. Mobile applications have also been recently used for business-critical uses which demand a more secure environment to operate on and to get developed. This situation has called for implementation of software development processes on blockchain or by using concepts of blockchain.

Challenges in Blockchain-Oriented Software Development

BOS or Blockchain-Oriented Software is a term used to describe a software which works with an implementation of a blockchain. The most important elements of the blockchain are-

i. Information excess as every node has a duplicate of the blockchain
ii. Exchanges are done dependent on public-key cryptography.
iii. Recording exchanges in consecutive blocks (done using consensus algorithm).
iv. Checking necessities of a procedure before validation.
v. Utilization of transaction scripting language.

The major challenges faced in BOS are described below:

1. NEW PROFESSIONAL ROLES

Usage of blockchain widely for development of software and applications to be run on mobile devices calls for extremely trained individuals from the IT sector. These individuals must have optimum knowledge of technology, finance and law if they must deal with blockchain development. They must also be trained in data-security to be proficient enough to tackle any potential attacks to blockchain as no technology is 100% safe.

2. SOFTWARE ARCHITECTURE

The most appropriate blockchain implementation is to be selected to suit the requirements of the desired product. Decisions need to be taken whether to use a side-chain mechanism or an ad-hoc blockchain mechanism in the implementation. E.g. Ethereum5 has embraced a basic database, a key-value store. By

ISSN:0971-1260

Vol-22-Issue-41-December-2019

embracing an information portrayal of higher level, for example, an Object Graph, it is conceivable to accelerate procedures and activities. This would be expensive if a key-value store is utilized.

3. MODELING LANGUAGES

The UML (Unified Modeling Language) diagrams made for regular software development must be modified to be used for blockchain implementation of software. Making entirely new UML diagrams also works equally well if the BOS specifications are crystal clear. The issue is that Use Case diagrams, Class diagrams and State diagrams cannot efficiently represent the BOS development environment. UML diagrams are favored over Use Case diagrams, Class diagrams and State diagrams because UML diagram consists all these diagrams together for a better understanding and explanation. The next diagram shown one the other side highlights the contents of a UML diagram.

Study for a Better BOS Development

A paper shows plan of Land Administration and Title Registration model based on blockchain Technology. Its discusses as for how blockchain innovation can be used and propose a model to perform testing for affirmation using Markov chain. Markov chain is an exceptional method to manage ensure the rightness of a structure by surveying that any of its practices is a model for a given property.

Bitcoin is most acclaimed cryptocurrency. Bitcoin is a blockchain oriented software which was highly noticed by both software experts and the masses. It empowers dependable exchanges with a decentralized administration mechanism regardless of whether there are unreliable users in the system. This is the element which aided Bitcoin stick out and it was conceivable simply because of the development of blockchain technology.

This paper proposes a Software Testing Life Cycle (STLC) to be a part of the SDLC for efficient development of software. This Testing Life Cycle is with respect to BOS projects for cryptocurrencies like Bitcoin. This STLC will test the product in all viewpoints.

Suggested Solution

As per recent surveys on GitHub, the most widely used programming languages among 193 retained repositories are JavaScript, C++, Ruby, Python and Go. These languages account for over 25% of the total repositories. Java repositories alone account to hardly 4% of the total. The repositories of cryptocurrencies like Bitcoin and Ethereum were made not more than 4 years back. All of them have a significant number of problems. Shown below, in Table 1, are the statistics for stargazers, contributors, open issues, watchers and forks for the top BOS repositories. The data in the table is as of September 23, 2016.

Based upon the figures of Table 1 and the facts stated above, I would hereby suggest some solutions to make BOS Engineering more effective and problem-free.

GitHub Repository	Stargazers	Contributors	Open Issues	Watchers	Forks	Language
bitcoin/bitcoin	9966	396	547	1211	4266	C++
ethereum/go-ethereum	2160	78	285	367	695	Go
dogecoin/dogecoin	1153	300	52	149	505	C++
ripple/rippled	1144	53	118	246	338	C++
coinbase/toshi	839	18	97	98	187	Ruby
ethereum/cpp-ethereum	723	89	212	196	270	C++

Table 1: Statistics Across Top Repositories

The testing of a software or its prototype shall be done by a fixed number of individuals. The test cases for a regular project are directly proportional to the number of team members. Although the total number of test cases are higher for an increased number of team members but the number of test cases per member is low. It is hard to determine what kind of response testing of a software on the blockchain would give. Testing is dealt in detail in this paper, thereby proposing a solution for proper testing of BOS.

ISSN:0971-1260

Vol-22-Issue-41-December-2019

The testing and debugging processes for individual programming languages need to be made better. Being distributed is the major characteristic feature of blockchain and therefore testing needs to be done in isolation. This would need the mocking of objects which can be simulated.

The most important solution to manage Blockchain-Oriented Software Development is the implementation of Smart Contract Development Environments (SCDEs). They are the blockchain-oriented versions of IDEs. They might turn out to be revolutionary for the building and diffusion of BOS. An IDE like this would create an environment that could streamline smart contract creation using specialized languages. E.g. Ethereum uses 'Solidity' which is a language for writing contacts.

Most of the testing techniques focus only on smart contracts, the testing solution proposed in this paper focusses on other aspects of testing too. It would have 4 phases, which are-

PHASE 1: SYSTEM OVERVIEW

In the first phase, there must be an involvement of testers so that they know about all components of development. They are to be divided into teams and assigned components which they are to be responsible for. A component map is then generated. The component map consists of all components and the sub-components of the system. This would also include all interfaces. A system component is generated from this map which has shortlisted components to pertain blockchain technology.

The shortlisted components are mapped to a component diagram. This also define the scope of testing. Once this is done, the whole team knows what is to be done and which member must perform what task.

PHASE 2: TEST DESIGN

An elaborate strategy should be contrived to a particular blockchain at this level. Key parts that should be utilized in the framework are recognized. The hyper ledger composer model is utilized to test the blockchain-situated programming. This stage may utilize its very own displaying language.

Hyper ledger composer incorporates an article situated displaying language that is utilized to characterize the area model for a business definition. Hyper ledger composer CTO document is made from the accompanying components:

1. A set of resource definitions, encompassing assets, transactions, participants, and events.
2. Optional import declarations that import resources from other namespaces.
3. A single namespace. All resource declarations within the file are implicitly in this namespace.

The CTO modeling language is tightly focused with just a few keywords. The model for your business network resides in a file that has a .cto file extension, and contains definitions for the following elements: namespace, resources, imports from other namespaces, as required.

The following steps are used to initiate and test the model:

- Testing the business network: Models act as blueprint for the desired application.
- The Asset and Participant registries: Resources are instantiated, and their instances are stored in the respective registries.

The transaction is implemented as a JavaScript function in the library module. This gets the model up and running quickly and performs three tasks-

- Creates instances of all the assets and participants from the model.
- Sets property value on the instances.
- Stores instances in respective registries.

Use case diagrams are also created on the side and it is checked if the requirements are met. The sequence flow diagrams helps us identify easily, any missing requirements or steps since there are all activities as a part of the sequence diagram.

ISSN:0971-1260

Vol-22-Issue-41-December-2019

4 of the most important things to be kept in mind while Non-Functional Requirements testing are-

- Being Agile
- Planning and judging what are most important
- Having an efficient hardware setup
- Documentation

PHASE 3: TEST PLANNING

In this phase, it is decided as to how every type of test is to be done with an estimate of the number of tests needed to be done at every level and to what extent. If a system isn't available for testing, a private blockchain must be set up for testing purpose. This is called alternative testing strategy.

The test tools need to be decided in this phase. Table 2 shown below highlights which method and tools are to be used in which type of testing-

Testing Level	**Methodology and Tools**
Unit Testing	Test Driven Development
System Testing	Blackbox testing to verify blocks, contracts and updation of values using JavaScript
Integration Testing	No value coins like - TestNet (useful for cryptocurrencies)
UI Testing	Automated tests using Selenium scripts

Table 2: Testing Levels with Methodologies and Tools

All the use cases which were designed back in the phase 2 are mapped to the cases mentioned in Table 2. The output of phase 3 is the final testing strategy and a precise documentation which consists test cases.

PHASE 4: TEST EXECUTION AND RESULT VERIFICATION

The final phase involves execution of all tests with methodology that has been documented earlier along with tools from phase 3.

The fundamental exercises that must be engaged upon in this stage are low level check and validation of blocks, exchanges and smart contracts. On the off chance that any outsider interfaces are utilized in the framework, they are tried as well. Additionally, issues with the User Interface are likewise searched for.

After all the testing is conducted successfully, the results are to be consolidated, evaluated and shared with the client for whom the firm has been developing software. If there are any issues, a bug report must be generated. This report must contain all the defects identities and a detailed test report sharing executions that passed and the ones that failed.

The prime output of the final phase of this testing process is the results and defects report which is shared with the developers for further processing. This is a repetitive/iterative process that would keep going on until some significant or critical errors are found or until when the system is shut down explicitly.

The figure (Figure 2) on the next page is an elaborate diagram for the Testing Lifecycle I have proposed in this paper.

CONCLUSION AND FUTURE WORK

There are innumerable challenges that blockchain developers are facing today but it is important to respond to those challenges strongly and use a technology like blockchain for our benefit. This can be best done by following a set of rules and practices to avoid issues that are usually faced while developing software on the blockchain. Blockchain demands changed professional roles, new modeling languages and other specialized metrics. On the analysis of 193 repositories (based on a survey) out of 1184, I proposed new ideas to facilitate the development process by studying the data obtained. The Blockchain-Oriented Testing Lifecycle has been proposed after analyzing the team sizes, tools and testing activities. A software to test BOS is another idea proposed in the system as a separate state-of-art Smart Contract Development Environments (SCDEs) for quick and reliable testing.

ISSN:0971-1260

Vol-22-Issue-41-December-2019

ACKNOWLEDGEMENT

I express my sincere regards to Prof. Kapil Nagwanshi for providing me an opportunity to explore and work on a topic as crucial as this. I am grateful to him for his patience, motivation and enthusiasm. His guidance helped me at all points of time during researching and writing this paper.

REFERENCES

[1] Akshay Kakadiya, "Block-Chain Oriented Software testing approach" in International Research Journal of Engineering and Technology (IRJET), Issue 12. vol. 4, LDRP Institute of Technology and Research, Gandhinagar, Gujart, pp. 1593–1597.

[2] Porru, Simone & Pinna, Andrea & Marchesi, Michele & Tonelli, Roberto. (2017). Blockchain-oriented Software Engineering

[3] Challenges and New Directions. H. Poor, An Introduction to Signal Detection and Estimation. New York: Springer-Verlag, 1985, ch. 4.

[4] Flavio Corradini, Giorgia Meschini, AlbertoPolzonetti, Oliviero Riganelli. "A Rule-Driven Business Process Design", 2007 29thInternational Conference on Information Technology Interfaces, 2007

[5] D. Larimer. Introducing bitshares object graph. [Online]. Available: https://goo.gl/TWWSif.

[6] M. Swan, Blockchain: Blueprint for a new economy. O'Reilly Media, Inc., 2015.

[7] Kombe, Cleverence & Manyilizu, Majuto & Mvuma, A. (2017). Design of Land Administration and Title Registration Model Based Blockchain Technology. Journal of Information Engineering and Applications. 7. 8-15.

[8] Unicredit, "Blockchain technology and applications from a financial perspective," 2016.

[9] G. Hileman, "State of blockchain q1 2016," 2016.

[10] Christoph Schmidt, Christian Diedrich. "Specification of test rules for vehicle comfort systems with formalized models and methods", 2006 IEEE International Conference on Industrial Informatics, 2006

[11] M. Aberdour, "Achieving quality in open-source software," IEEE software, vol. 24, no. 1, pp. 58–64, 2007.

[12] Testing of Blockchain: http://www.irjet.net/.

[13] Simone Porru et.al, 2017. "Blockchain-oriented Software Engineering: Challenges and New Directions", 39th IEEE international conference on Software Engineering Companion.

[14] https://steemit.com/business/@divyanthj/is-it-possible-to-put-the-software-development-lifecycle-sdlc-on-the-blockchain.