International Journal of
INTELLIGENT SYSTEMS AND APPLICATIONS IN ENGINEERING
ISSN:2147-6799 www.ijisae.org Original Research Paper

A Framework for Software Testing of Blockchain Based Applications with Appropriate Tools

Smita Bansod*[1], Lata Ragha[2]

Submitted: 27/01/2024 **Revised**: 05/03/2024 **Accepted**: 13/03/2024

Abstract: Blockchain technology is becoming the preferred technology for various applications in the digital world notwithstanding it being an emerging technology with a few grey areas relating to standards, best practices, and vulnerability to attacks. Due to the inherent features of blockchain, there are serious challenges in deciding on an appropriate testing framework and selecting the tools for the rigorous testing, validation, and verification of the software being developed for the evolving blockchain based applications to ensure glitch free operations in a practical working environment. The software testing needs and parameters to be tested have been discussed in detail and a testing framework has been proposed keeping in mind the special needs of blockchain based applications, such as smart contract testing, type wise testing and layer wise testing. A study of the various testing tools available have been made and a comparison table presented to identify an appropriate testing tool for specific applications.

Keywords: *Blockchain Technology, Software Testing, Quality Assurance, Testing Framework, Automation tools*.

1. Introduction

Ever since the introduction of Blockchain Technology in 2009 [1], this technology has been attracting attention and today it is the most preferred technology for various applications. However, hurdles like vulnerability to various types of attacks come in the way of Blockchain becoming the preeminent technology of the digital world. International Standards like IEEE and NIST are yet to agree on a single definition of Blockchain. While the business due to Blockchain applications has been predicted (by Gartner) to exceed $3.1 trillion by the year 2030, the applications themselves are in preliminary stages of development, with several issues and glitches relating to platforms, languages, consensus mechanisms, etc., leading to attacks which will result in collapse of the system and data loss. Paper [2] discusses the details of various applications of Blockchain technology and their vulnerability to attacks.

To facilitate the large-scale adoption of Blockchain based applications, it is necessary to make these applications robust with glitch free operation conforming to accepted standards. There is a need for rigorous software testing approach to protect, verify and validate the applications based on blockchain technology. Similar to established standards like Object-oriented Testing, Web based Testing, Agile testing and so on to ensure quality of software, there should be well defined framework for testing the software of the various applications based on blockchain to ensure quality. Being an evolving technology, there are serious challenges in testing blockchain technology due to the absence of established best practices, the generation of appropriate test data and issues relating to scaling, security, and performance.

"Quality" refers to the overall measure of how well a software application or system meets its intended requirements and performs its functions effectively and reliably. Software quality is a complex concept which defines Functionality, Reliability, Performance, Usability, Security, Compatibility, Maintainability, Testability, Scalability, Robustness, and Interoperability. The systematic evaluation of these quality criteria is done through software testing to identify and rectify the gaps and resolve the issues. Numerous researchers are working to arrive at an appropriate framework of software testing to ensure glitch free operations of the applications based on blockchain technology [3], [4]. The methodology, methods, and techniques of such a systematic approach to software testing will go a long way in resolving the quality issues relating to security, performance, and smart contract.

The flow of the paper is explained in the following sections: Section 2. General discussion on Software Testing and Quality Assurance. Section 3. Brief explanation of blockchain technology, development of applications and special requirements of blockchain based applications / software. Section 4. Discussion on the research work and the limitations. Explore the use of Blockchain technology for software testing and testing for blockchain based software as a part of literature survey. Section 5. Challenges in software testing for blockchain based applications. Section 6. Proposed software testing framework specifically for blockchain based applications which results in ease of

[1] *Faculty, Shah & Anchor Engineering College and Research Scholar, Terna Engineering College, Mumbai University, Mumbai, MH, India*
ORCID ID : 0000-0002-7057-9754
[2] *Fr. C. Rodrigues Institute of Technology, Vashi, Mumbai University, Navi Mumbai, Maharashtra, India*
ORCID ID : 0000-0002-6055-6671
** Corresponding Author Email: sakec.smitab@gmail.com*

testing. Section 7. The software testing support tools - their availability and selection for Blockchain based applications.

2. Software Testing and Quality Assurance

The goal of providing quality assured software is achieved by (i) detecting and eliminating bugs, (ii) providing a reliable system with minimum risk (iii) reducing maintenance cost, and (iv) refining the process for future testing. Also, the product should meet the requirements of the intended application.

The figure 1 explains the basic software testing model which must include the development software as per the requirements, the bug model, testing methodology and trained testers who are knowledgeable in the field of software testing and the technology used [5].

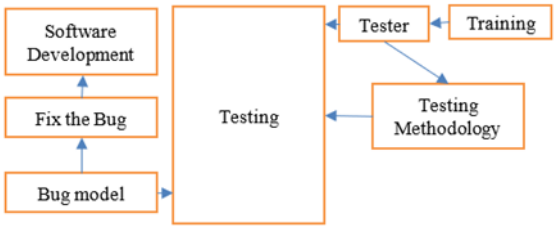

Fig 1. Basic Software Testing Model

Once the operational requirements are firmed up, the testing process should be taken up in close collaboration with the software development team. The test management team has to draw up a detailed testing plan taking into account the testing process, details of test cases with specifications, the test environment, the execution of test cases, monitoring their assessment after execution, tracking the defects, identifying the testing tools [6], and the test reporting before arriving at the quality assurance of the product.

The software development goes through several stages after the requirements are firmed up. The fault or bug detection is a lengthy process shown by figure 2. Any deficiency in the early phases like requirement gathering, analysis and design phases must be identified and corrected early. Identifying and fixing the faults after implementation will lead to high rectification costs. 'Fault in' phase consists of requirement gathering and analysis, designing and coding phases. Under the 'Fault out' phase, action is taken to classify the detected faults, assign priorities and resolve the faults as per the assigned priority sequence.

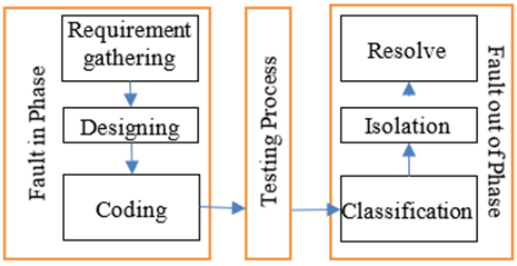

Fig 2. Fault Detection Model

The detection of bugs/ faults must be done using a proper methodology [7]. The software testing methodology is based on the testing strategy which consists of test factors, test phases and testing approaches. The strategies adopted for Verification and Validation of software differ from each other. Validation strategy is used to check whether the developer has built the right product conforming to the industry standards following the various testing milestones at the unit, integration, functionality, system, acceptance, and installation testing stages. Verification testing is the process to check whether the developer has built the software with the correct data to meet the requirements of the client, validated through different steps for requirement gathering, requirement specification, functional design (high level), internal design (low level) and coding. Separate tools are required for testing in Static and Dynamic modes. In Static testing, the project is verified by review committee members without execution. In Dynamic testing, the software is tested by actual execution to detect faults using white box testing or black box testing. Testing tools are tools used for various activities to automate the testing process like management of overall testing process, verification and validation testing and generating reports for analysis. The structure and the code are checked using white box testing followed by black box testing with focus on input and excepted results.

Based on the tests, all gaps and faults detected in the software are fixed by the developer and then the software is subjected to regression testing to verify that the fault rectification patches do not affect the other modules in the software. While managing the software testing process, the aim is to test the high priority test cases ahead of the others to improve the testing efficiency by minimizing the test suite.

3. Blockchain Framework and Importance of Testing

Blockchain is an emerging technology with features such as immutability, sequential chain of cryptographic links in distributed ledger architecture with validated blocks using consensus algorithms. This technology is quite attractive for secure applications in various domains. Blockchain evolution goes through three phases: 1- Cryptocurrency, 2- Ethereum with blockchain and 3- Applications development in different areas/domains.

A system to prevent tampering with the timestamps of digital documents using distributed computing was introduced in the 1990's paper [8]. The idea was to provide a secure and tamper evident way of recording information. Later, Satoshi Nakamoto created Bitcoin [1] and put forward the concept of blockchain technology with decentralized ledger maintained by anonymous consensus. The bitcoin cryptocurrency was deployed in application

related to cash and first bitcoin transaction was recorded in May 2010 for buying two pizzas in exchange of 10000 bitcoins. This was followed by various cryptocurrencies which were introduced and digital payment system was booming during the period 2012-13. Vitalik Buterin was the pioneer who introduced the smart contract platform-based application in the year 2013 when he was just 19 years of age. The Hyperledger project launched in the year 2015 by the Linux Foundation introduced the concept of 'Permissioned blockchain' network. This concept was the turning point in the development of blockchain technology, which opened the technology for various applications. After 2016, the blockchain technology was moving very fast on its own with many researchers working to make it a fool proof technology with rigorous testing strategy to make the applications robust and glitch free. The timeline of the blockchain technology is shown in figure 3.

Fig. 3. History of Blockchain Technology

Blockchain Architecture (shown in figure 4) is crucial to understand the technology. A chain of blocks is formed by connecting several validated blocks. Each block contains block version, Root hash value, timestamp of validation, difficulty level in n bits, Nonce as validation matching value, Parent block hash value and various transactions hash value. Blockchain network consists of various nodes which are connected virtually with each other in a mesh configration. The nodes have complete data of blockchain based on their types. Basically, Full node has all the data of blockchain while Light node has its own transaction related information.

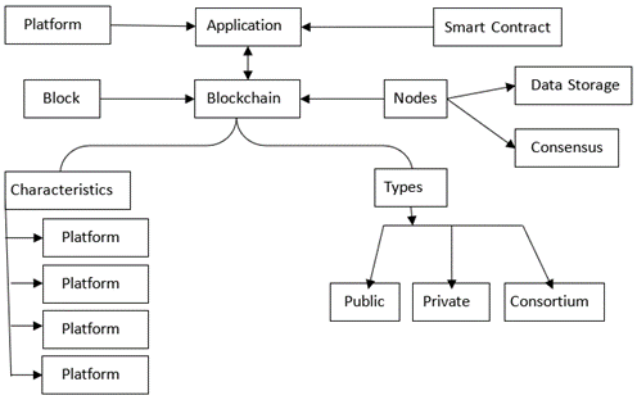

Fig. 4. Blockchain Architecture

Blockchain technology basically supports key features such as Decentralization, Persistency, Transparency and Auditability. These features should be verified at the testing stages to give concrete support to the application and satisfy the requirements.

The strategy for testing the two primary categories of Blockchain – Public blockchain and Private blockchain – should be tailored suitably to meet the requirements of each category. In the case of a public blockchain, emphasis should be placed on aspects such as network security, vulnerabilities in smart contracts, and wallet security. On the other hand, for a private blockchain, testing focus should be directed towards component-level tests, smart contracts, identity management, infrastructure security, and related areas. Consortium blockchain is the third type, which is a combination of both public and private blockchains. The testing strategy for this category needs to be designed taking into consideration the structured data which could be for both private and public networks.

Blockchain blocks are validated by miners using calculation algorithms i.e., consensus algorithms. The consensus algorithms need to be thoroughly verified and validated at the design stage itself to ensure that the applications will be glitch free. There is a list of consensus algorithms available and further research is continuing to improve the performance and speed up the operations.

The block is validated by calculating the hash values of complete block using nonce to satisfy n bits challenge. The Merkle tree algorithm is used to calculate the transaction or data hash value in block. The actual data needs to be stored on some virtual decentralized data storage like IPFS or Swarm and then connected to blockchain network using hash values in blocks.

The application development happens using platforms like Ethereum, Hyperledger etc. If the platform is vulnerable, then the application built on it is also vulnerable. Hence selection of platform must be done carefully taking into consideration the special requirements of each application.

The testing of smart contract must be done meticulously, as this is the heart of the application. The history of blockchain indicates that the earlier failures and attacks happened due to lack of knowledge about smart contract writing, language used and so on. The systematic study [9] discusses the identified issues relating to smart contract and solutions for the same. The validation and verification of smart contract by experts at the testing stage should be done carefully and on priority considering the real operation of the application. Once the smart contract is deployed, making changes is cumbersome and costly. In the past, attackers had found vulnerable networks and caused huge losses through DAO and other attacks.

4. Testing Frameworks for Blockchain Based Software - Literature

The literature survey starts with a discussion on the testing of software using Blockchain technology but later focusses on the testing of software meant for applications based on blockchain technology in Table 1. This paper proposes a

framework for testing the software of blockchain based applications with appropriate tools.

Blockchain technology is used to collaboratively test the software developed for major projects in a secure way. The management of the testing teams working on development of large and elaborate software systems is a challenging task. A collaborative system of software testing using blockchain technology has been proposed in [10]. This collaborative model resolves most of the hurdles posed by integration issues of blockchain based software testing.

The Agile Plus framework was able to overcome the deficiencies of the earlier software development projects [11]. Software testing is performed through various layers and all the transactional steps with data including smart contract execution are stored in a private Ethereum blockchain and IPFS respectively for ensuring that the system remains tamper-proof with reduced risk of hacking. The improvement in the engineering quality and testing method of scrum based blockchain development projects is discussed in paper [12]. The sustainable life support provided by blockchain through sprints in the development of BOS (Blockchain Oriented Software) is explained in paper [13]. This has ensured customer satisfaction throughout the life of the software.

Many researchers have introduced various tools and frameworks for blockchain based testing. One of them is by Kaya [14] , which facilitates quick formulation of test cases for front-end behaviour and back-end logic. This tool assures about the set the blockchain pre-states, and generate readable analysis reports. It is basically used for automated analysis and report generation of smart contract by validation method. A few platforms are covered with the help of this framework and research is on to enhance its adoption.

Coverage of the applications through Test cases is an important factor in testing of the software. As discussed previously, smart contract is the heart of blockchain based applications. By meticulous testing of the smart contract implementation, the applications can be made more safe and secure. Control Flow Graph (CFG) is used for data flow testing with higher coverage of test cases by enhanced Genetic Algorithm [15]. This framework is basically consisting of the following three steps: CFG construction, data flow analysis, and test-case generation. The framework is efficiently generating high priority / quality test cases. Also, there is improvement in performance, execution, and iteration time of the random test cases.

Smart contract is combined with various other entities such as User, Hackers, Transactions, Mining, Block and then Registered contract. While verifying the smart contract and behaviour of the model, all the related entities should be verified simultaneously to check their interaction with the smart contract. The formal verification model [16] verifies the statements and logic of the contract and checks with other entities for integration of the complete system for proper testing and smooth working of the applications.

Smart contract verification, validation and testing using public blockchain test network (like Ropston, Rinkeby), Security analysis tools (Oyente, Securify), Emulators (Hyperledger Calliper, Hardhat, Ganache), and Simulator (Gauntlet) has been suggested in [17]. Further, analysis must be carried out on smart contract by its well-known vulnerabilities such as Smart contract source code, application interaction with other code or smart contract and blockchain system transactions and nodes creation vulnerabilities. The detection of vulnerabilities of smart contract and their comparison using different methods like formal verification method, fuzzy testing method, stain analysis method, symbol execution method, intermediate representation method and deep learning method have been discussed in [18].

Smart contract testing is conducted using Mutation testing method named as SuMo (SOlidity MUtator) [19]. Analysis of the Solidity Documentation and existing tools led to eleven innovative mutation operators, with seven of them specifically focusing on the distinctive features of Solidity. This method is faster and is very effective in detecting test cases efficiently.

Different researchers have focused on various factors. Latency, throughput, response time are calculated and test cases are designed using Ethereum test network [20]. The performance of Hyperledger fabric blockchain framework is analysed using the factors throughput, latency, and scalability [21].

5. Challenges in Blockchain Testing

Table 1. Testing areas and their limitations

Reference Paper	Focus (Application/ Network/ Smart contract)	Testing for / using blockchain	Testing area	Limitation
[10]	Application	Using	Large and complex application integration	The substantial costs associated with the storage, computation, and setup of a blockchain architecture.
[11]	Application	Using	Improves security, traceability, communication, coordination transparency, and improves mutual trust between customer and developer.	User rating and review, translator for language support, task distribution in developer team also.
[12]	Application	Using	Improve software quality, development efficiency, leading to efficient project management O&M services.	Missing practical approach between software developers and project managers.
[13]	Application	Using	With appropriate monitoring matrix, sustainability is ensured.	Focus on supply chain sector only. Further domains need to explore.
[14]	Smart Contract	For	Ease the creation of test cases, offering a flexible and convenient method for test engineers to establish blockchain pre-states and generate meaningful reports.	Do not support to variety of blockchain platforms.
[15]	Smart Contract	For	With smaller number of iterations and less execution time, high coverage of the test cases is generated.	Applying advanced fitness function, large and complex smart contract cases can be tested.
[16]	Smart Contract	For	Formal verification method using syntax, models, and logic.	Do not support more advanced smart contract syntax.
[17]	Smart Contract and network	For	Comparison based on public test networks, security analysis tools, blockchain emulators and blockchain simulators are compared with the support of smart contract.	The survey, no proposal for better solution to overcome the limitation.
[18]	Smart Contract	For	These are meant for smart contracts. For high accuracy results the static and dynamic detection methods are to be combined.	Combination of static and dynamic detection methods will enable the detection of more types of vulnerabilities of multi-version smart contracts.
[19]	Smart Contract	For	A reliable Solidity code is delivered and the fault detection capabilities are improved with this Tool for Ethereum blockchain.	Expanded SuMo with automatic regression mutation are needed to identify the nature of such patterns and to understand whether they overlap with some existing operators.

Absence of Best Practices: One of the significant hurdles in the Blockchain realm stems from a deficiency in skills or experience in developing Blockchain applications. Acquiring additional skills or grasping the best practices for implementing Blockchain applications can be quite costly.

Deficiency in Blockchain Testing Tools: The crux of successful Blockchain implementation lies in utilizing the right tools, particularly when testing Blockchain-based applications/products. Without the appropriate toolset, the likelihood of failure increases. Effective testing of Blockchain applications demands a diverse toolset, focusing on Bitcoin and encompassing various Ethereum tools.

Lack of Standardization in Blockchain Testing: Beyond technical proficiency, possessing legal expertise is crucial for fostering Blockchain adoption. Limited awareness of the complex blockchain concepts amongst the developers is the major cause for the numerous issues in the Blockchain space.

Suboptimal Test Strategy: Testing is often relegated to a lesser role than programming, resulting in the development of a Blockchain application environment with either minimal or no dedicated testers for exploring and evaluating Blockchain products. The suboptimal testing strategy leads to inefficient testing of Blockchain applications, either through repetitive testing or, in some cases, no testing at all.

Irreversibility of Transactions: Implementing Blockchain without due diligence exposes organizations or users to high asset risk, as Blockchain transactions are irreversible. Establishing controls to prevent redundancy and ensure additional safety poses a serious challenge.

Block and Chain Size: In addition to having standard testing practices, tools, and protocols in place, it is crucial to validate the block and chain size to avoid failure of applications.

Performance and Load: Proper performance and load testing in both the development phase and under different network conditions will ensure glitch free operations in actual live environment.

Consistency/Availability: In Blockchain applications, achieving consistency in Bitcoin does not happen simultaneously with availability and partition tolerance. Transitioning to strong consistency often presents a myriad of challenges.

Security: Proper security check at the testing stage will reduce vulnerability of applications to attacks at the network, user, and mining levels.

Time consumption for validity: Transaction and block validation consume time because of consensus algorithm complexity. New arrival of consensus algorithm testing and report generation requires a lot of time to rectify fault in it. Also, compatibility testing of consensus algorithm poses a real challenge.

6. Proposed Blockchain Based Testing Framework

Figure 5 below shows the functional block schematic of the proposed blockchain based software testing framework.

Fig. 5. Software Testing Framework

6.1. Initiation

Software Quality Assurance and Control team initiates the testing process after fully understanding the requirements of the application and the technology adopted.

6.1.1. Generate and Maintain Test Policies and Standards

Team members contribute in policy making and the manager takes the decision on the quality of the product. Industry standards should be followed by the organization while developing the software to assure the quality to customers. The Quality aspect has to be kept in view throughout the testing process and the policies and standards should be maintained consistently.

6.1.2. High Level Test Phase Plan and Resource Scheduling

General project data is collected from the customer and software engineering team. This data is analyzed and a master plan is drawn up for resources planning and scheduling.

6.2. Test Planning

Testing plan is drawn up conforming to the master plan and scheduling. Team members and the related managers actively participate in the requirement design and coding activities and collect the information. The testing team along with SQA team draw up the high-level plan and decide the major objectives to verify and validate the developed software product.

6.2.1. Sympathetic/ understanding Blockchain architecture

Blockchain architecture – platform, consensus, network, languages used in the application should be thoroughly understood to write and execute the test cases based on it.

6.2.2. High level planning

The high-level plan for testing objectives and resource training scheduling should be decided. Also, the project testing life cycle and milestones should be clearly identified.

6.2.3. Test Design

Detailed test design for the validation task involves aligning the requirements or features with the specific test cases intended for execution with the expected outcome using test oracle. The best tools and environment for the execution of the test cases have to be selected.

6.2.4. Detailed test case writing

In accordance with the initially opted standard, test cases should be specified (e.g., IEEE recommended test specification with components) in detail which should include the purpose, the need, the requirement of special environment, procedure for execution, inter-case dependency, intra-case dependency, output specification, etc.

6.2.5. Test Data Creation

Test cases execution and verification requires data in the form of input as well as expected output. Input data are generated using different technologies. One of these is boundary value analysis which is a type of block box testing. Test oracles are used to generate expected output data. While designing test cases, all data should be generated, so that the execution is simplified.

6.2.6. Environment setup

Test cases should be executed in a similar environment as the application is expected to face in real time deployment. As the Blockchain platform related testing environment is cumbersome, adequate advance action is required to train the testing team to work in the required environment.

6.2.7. Consideration of Measurement and Matrices

While designing test cases, various measurements and matrices required to identify the performance of each component as well as the total application must be factored into the testing process to ensure that nothing is left out.

6.3. Testing of Blockchain based System

Unlike testing of traditional applications, Blockchain based applications need to be tested based on blockchain features, interfacing with other software, platforms, networks, components, types and layers of blockchain architecture. Hence the testing is carried out in two parts: testing of types and testing of layers.

6.3.1. Type wise testing

Functional Testing: Basic functionalities of the various components of blockchain are examined. A new block can be added only when the validity of the transaction is confirmed. Assessing block size, chain size, verification of a block before it is added are some of the examples of testing

Integration Testing: Given the involvement of multiple components in a blockchain application, it is crucial to conduct regular and thorough integration tests to confirm the proper integration of all components.

Security Testing: Security testing is indispensable for debugging blockchain applications, particularly in environments where high levels of security are paramount, such as in financial, government, or regulatory settings.

Performance Testing: The speed of blockchain applications is a critical factor, and performance testing evaluates the speed of executing the transactions within the network.

Node Testing: The consensus achieved among all the nodes determines the sequence of adding and storing transactions. The consensus protocol decides the strength of a blockchain.

Smart Contracts Testing: Smart contracts are software modules on the blockchain that automatically execute transactions. Testing smart contracts involves ensuring that parties involved in transactions adhere to the specified rules.

API Testing: API testing evaluates the interaction of applications within and outside the blockchain system.

Cycle Testing: It ensures the durability of a blockchain throughout every transaction cycle, guaranteeing the orderly storage of added transactions within the blockchain network.

Interoperability Testing: If the system architecture includes various platforms and other blockchain networks, then the cross blockchain transactions must be tested using interoperability testing.

Data Immutability Testing: It ensures that data cannot be altered or deleted, once it is added to the blockchain.

Consistency and Atomicity Testing: It ensures that all the changes are either applied or none will reflect on the blockchain while transactions take place.

Fault Tolerance and Recovery Testing: In the event of failures like node crash or network partition, the system should recover immediately.

Regulatory Compliance Testing: It ensures that the blockchain application complies with relevant regulations and legal requirements.

Privacy Testing: It ensures that sensitive information is appropriately protected [22].

Penetration Testing: This test identifies potential security vulnerabilities which must be addressed.

Bug bounty: Development of 100% bug free software is not possible. So, organizations encourage security researchers and ethical hackers to use and report vulnerabilities in their software by offering monetary rewards. It will be helpful to organizations to enhance the security level of the software by leveraging the skills of the global community of security researchers.

Blockchain layered architecture and the related vulnerabilities are mentioned in [23] which are helpful to test against those vulnerabilities. Also [4] discusses about layer wise vulnerabilities related to Smart Contract, Performance, and Security of blockchain applications and test automation using various tools.

6.3.2. Layer wise testing

Data layer: Data layer testing ensures that transaction data, hash values and wallet data are properly encrypted with appropriate cryptosystem to be quantum resistant and consistent.

Network layer: Interconnectivity, authentication for users and providers of network services in order to protect using redundancy, availability, routing, etc, to be tested thoroughly.

Consensus layer: Consensus algorithms are to be tested to ensure to realise all the advantages.

Replicated state machine layer (Smart Contract and transactions): User identity, transaction data confidentiality, integrity, authentication, and availability should be tested and validated using safe language, static/ dynamic analysis, formal verification, and audit.

Application layer: User interface, malware, application developers, token issuer, regulatory requirements must be tested for multi factor authentication, decentralized authority, and application-level privacy preservation.

6.4. Test Report

After completion of the tests, the report is generated which

is a summary of all the tests cases executed with the evaluation results. The report is useful for maintenance of the system as well as to plan the test cases of similar kind of blockchain based products.

6.4.1. Defect Tracking

On testing, if a failed status is indicated, the defect or gap should be fixed according to its impact and severity level. The side effects of fixed defect also need to be checked. In case a defect is detected but could not be fixed, then that particular defect has to be kept under observation throughout the life of the software.

6.4.2. Test Report and Analysis

This provides a summary of the overall product like component details, project dates, costs, task details, test results, action taken and future scope. This report will form the basis for the maintenance of the product and for future actions.

6.4.3. Smart Contract testing reports

This is an important report for future blockchain based projects and to consider evaluation measures and matrix for future test planning. This report describes the details of Smart contract, data, and rules for processing.

6.5. Test Measurement and Monitoring

In blockchain-based projects, various measurements and metrics are used to assess and improve aspects like performance, security, and throughput of the system. Monitoring in a blockchain-based project refers to the process of observing and analyzing various aspects of the blockchain network and its associated components to ensure its health, security, and optimal performance. Monitoring is crucial for maintaining the integrity of the blockchain, and ensuring that the network operates according to its design objectives and requirements by identifying and rectifying potential issues. Effective monitoring in a blockchain-based project contributes to the overall reliability, security, and performance of the network, ultimately enhancing the user experience and trust in the system.

6.5.1. Key Measurement Parameters

Some of the key parameters that are measured in blockchain systems are discussed below.

Throughput /Transactions Per Second (TPS): Higher TPS values are desirable for scalable and efficient systems. This metric measures the number of transactions processed by the blockchain network in one second.

Latency / Transaction Confirmation Time: Lower confirmation times are preferred for better user experience. Time taken for a transaction to be confirmed and added to the blockchain.

Scalability:

• Network Scalability: The network should be able to handle increase in the number of nodes and transactions without deterioration in performance.

• Vertical Scalability: Capacity of individual components and nodes within the system are increased.

• Horizontal Scalability: Distribution of load which adding more nodes should be assessed.

Security:

• Hash Power (applicable to Proof-of-Work): The computational power contributed by miners to secure the network.

• Consensus Algorithm Security: The security features used (e.g., proof-of-work, proof-of-stake) should be assessed.

Decentralization / Node Distribution: Examining how nodes are distributed across the network to ensure a decentralized and resilient system.

Interoperability / Cross-Chain Transactions: Measuring the ability of the blockchain to interact with other blockchains, facilitating interoperability should be assessed.

Smart Contract Execution / Gas Fees: In platforms like Ethereum, gas fees are essential metrics to measure the cost of executing smart contracts.

Governance / Participation Rate: The percentage of stakeholders participating in governance decisions should be factored.

Consensus Efficiency / Finality Time: The time taken for a block to be considered irreversible should be measured.

Data Storage and Retrieval:

• Storage Costs: Evaluating the cost of storing data on the blockchain.

• Data Retrieval Time: Measuring the time it takes to access stored data.

Adoption Metrics:

• User Base: The number of users and entities who are active participants of the network.

• DApp (Decentralized Application) Usage: Monitoring the usage and popularity of decentralized applications built on the blockchain.

Energy Efficiency / Energy Consumption: Particularly relevant for proof-of-work blockchains, measuring the environmental impact of mining activities.

6.5.2. Key Monitoring parameters

Some of the key monitoring parameters for blockchain based systems are discussed below.

Node Health:

• Monitoring the health of individual nodes in the blockchain network is essential. This involves checking for issues such as hardware failures, connectivity issues, and software malfunctions.

• Node health monitoring helps ensure that all nodes are operational, contributing to the consensus process, and maintaining the integrity of the blockchain.

Consensus Mechanism: Monitoring the consensus mechanism is critical for ensuring that the agreed-upon rules for validating transactions are followed. This includes monitoring the block creation process and confirming that consensus is achieved among nodes.

Network Performance:

• The potential bottlenecks, latency issues, or network congestion issues which have profound impact on the speed and efficiency of transactions will come to light with this testing.

• Analyzing network performance data can assist in optimizing the blockchain's infrastructure to handle increasing transaction volumes.

Security Monitoring:

• Continuous monitoring for potential security threats and vulnerabilities is crucial. All suspicious and malicious activities that may compromise the confidentiality, integrity, or availability of the blockchain network will be identified.

• Security monitoring may involve the use of intrusion detection systems, audit logs, and other tools to detect and plan mitigation.

Smart Contract Execution: If the blockchain project involves smart contracts, monitoring their execution is vital. This includes tracking the performance of smart contracts, identifying any errors or vulnerabilities, and ensuring that they operate as intended.

Transaction Monitoring: Monitoring transactions in real-time helps ensure that they are processed correctly and efficiently. This involves tracking transaction confirmations, validating transaction inputs and outputs, and identifying any irregularities or errors.

Resource Utilization: Monitoring the usage of Processor and storage helps ensure that the blockchain nodes and associated infrastructure have sufficient resources to operate efficiently. This is particularly important in large and decentralized networks.

Scalability Monitoring: As the blockchain network grows, monitoring its scalability becomes crucial. This involves assessing the network's ability to handle increased transaction volumes and user activity.

Alerting and Notification: Implementing alerting systems that notify administrators or relevant stakeholders in real-time when predefined thresholds or anomalies are detected. This allows for prompt responses to potential issues.

Compliance Monitoring: Ensuring compliance with regulatory requirements and industry standards by monitoring and documenting relevant activities within the blockchain network.

7. Choice of Appropriate Blockchain Based Testing Tools

Automation testing tools reduce the efforts of both static and dynamic testing by reducing human involvement and errors. Automated testing combined with regression testing leads to reduced overall cost of software development. Blockchain technology-based applications need some special testing tools to verify and validate the system. Testing tools need to be carefully selected based on the purpose, software testing life cycle, tester skills/knowledge, affordability of that/those tool. Table 2 provides a comparison of the various tools available and this table could serve as a guide for the choice of the tool based on the purpose, features, advantages, and disadvantages.

8. Conclusion

Software testing is essential to assess the functionality of the program and to check that the software is free from gaps, bugs, and errors ready to deliver the expected outcome in a live environment. Such testing of software for Blockchain based applications imposes special requirements and the use of appropriate tools for testing the various parameters results in ensuring the quality of software at optimum cost. Researchers are refining the various approaches to software testing using sophisticated testing tools for blockchain based applications to ensure glitch free operation for the benefit of the users. Further, there are indications that in future it would be possible to test software using blockchain technology.

Table 2. Comparison of Blockchain based software testing tools

Tool	Purpose	Features	Advantages	Disadvantages
Truffle Suite [24]	This is a well-known Ethereum development and testing framework.	* Testing and deployment of Smart contract. * Automated testing with Mocha and Chai. * Built-in support for Ethereum Virtual Machine (EVM).	* Comprehensive development and testing framework. * Supports Ethereum and other blockchain networks. * Integrated with Ganache, a personal blockchain for testing. * Simplifies smart contract development with a built-in deployment system.	* Limited support for non-Ethereum blockchains. * Can be complex for beginners.
Ganache [25]	Ganache is a personal blockchain for Ethereum that can be used to deploy contracts, develop DApps, and run tests.	* Local blockchain environment. * Gas tracking for transactions. * Quick development and testing.	* Local, private Ethereum blockchain for testing. * Quick and easy to set up. * Supports account management, gas control, and snapshot features.	* Limited to Ethereum-based projects. * May not fully simulates the behaviour of a public blockchain.
Remix [26]	Written in Solidity language this is a open-source web and desktop application that helps in smart contract development,	* Integrated Solidity compiler. * Debugger for debugging smart contracts. * Gas estimation and deployment tools.	* Has the ability to switch between different networks with user-friendly interface. Supports various plug-ins and extensions.	* New developers will face a steep learning curve to understand decentralised technologies and understand smart contract. * Some IDEs may be designed for specific blockchain platforms, limiting cross-platform compatibility.
MythX [27]	This helps in identifying and fixing security vulnerabilities in Ethereum smart contracts.	* Security analysis tools. * Integration with various development environments. * Continuous security monitoring.	* Security analysis platform for Ethereum smart contracts. * Integrates with various development environments and CI/CD pipelines. * Provides in-depth security reports and suggestions.	* Subscription-based pricing model for advanced features. * Some features may be more suitable for professional development teams.
Solhint [28]	These are linters for Solidity, the programming language for Ethereum smart contracts to ensure code quality and adherence to best practices.	* Static code analysis for Solidity. * Customizable rules and configurations.	* Linter for Solidity, the programming language for Ethereum smart contracts. * Helps identify and fix code quality issues early in the development process. * Integrates with popular development environments.	* Focuses on static analysis, may not catch all runtime issues. * Limited to Solidity-based projects.
Hardhat [29]	This has a built-in testing framework suitable for development phase. It helps debug Ethereum software.	* Extensible plugin system. * Support for TypeScript. * Scriptable tasks for automation	* Flexible and extensible development and testing environment. * Supports Ethereum and other EVM-compatible blockchains. * Built-in functionality for smart contract testing and deployment.	* Less documentation compared to some other tools. * Steeper learning curve for beginners.
Hyperledger Caliper [30]	This is a tool for benchmarking blockchain technologies.	* Benchmarking multiple blockchain platforms. * Configurable workloads.	* This is an open-source project providing metrics for assessing performance. * Caliper supports multiple blockchain platforms, including Hyperledger Fabric, Sawtooth, and others.	* Configuring Caliper for specific blockchain platforms and scenarios can be complex, especially for users who are not familiar with the intricacies of blockchain networks. * While Caliper supports multiple Hyperledger projects, its primary focus is on benchmarking Hyperledger blockchains.
Cucumber [31]	Cucumber is a tool for running automated tests written in a natural language style. It is often used for testing smart contracts and decentralized applications.	* Behaviour-driven development (BDD) testing. * Test scenarios written in plain text.	* Behaviour-Driven Development (BDD) tool for testing. * Supports multiple programming languages. * Enhances collaboration between developers and non-developers.	* Requires well-defined specifications, which may be challenging for blockchain projects. * May not be as specialized for blockchain testing as other tools.

Acknowledgements

We thank Mr. Ramani Iyer for his valuable discussions and comments that greatly improved the manuscript.

Author contributions

Smita Bansod: Conceptualization, Methodology, Software, Data curation, Writing-Original draft preparation,

Lata Ragha: Visualization, Investigation, Validation.

Conflicts of interest

The authors declare no conflicts of interest.

References

[1] S. Nakamoto, "Bitcoin: A Peer-to-Peer Electronic Cash System," p. 9, 2008.

[2] S. Bansod and L. Ragha, "Blockchain Technology: Applications and Research Challenges," in 2020 International Conference for Emerging Technology (INCET), Belgaum, India: IEEE, Jun. 2020, pp. 1–6. doi: 10.1109/INCET49848.2020.9154065.

[3] Afsal Backer, "Blockchain testing." [Online]. Available: https://blog.testproject.io/2022/02/15/blockchain-testing/

[4] C. Lal and D. Marijan, "Blockchain Testing: Challenges, Techniques, and Research Directions." arXiv, Mar. 18, 2021. Accessed: Nov. 24, 2023. [Online]. Available: http://arxiv.org/abs/2103.10074

[5] "ISO/IEC/IEEE International Standard - Software and systems engineering --Software testing --Part 1:General concepts," IEEE. doi: 10.1109/IEEESTD.2022.9698145.

[6] K. M. Mustafa, R. E. Al-Qutaish, and M. I. Muhairat, "Classification of Software Testing Tools Based on the Software Testing Methods," in 2009 Second International Conference on Computer and Electrical Engineering, Dubai, UAE: IEEE, 2009, pp. 229–233. doi: 10.1109/ICCEE.2009.9.

[7] Naresh Chauhan, Software testing : principles and practices. Oxford University Press, 2010.

[8] S. Haber and W. S. Stornetta, "How to Time-Stamp a Digital Document," no. 3, pp. 99–111, doi: https://doi.org/10.1007/BF00196791.

[9] D. Macrinici, C. Cartofeanu, and S. Gao, "Smart contract applications within blockchain technology: A systematic mapping study," Telematics and Informatics, vol. 35, no. 8, pp. 2337–2354, Dec. 2018, doi: 10.1016/j.tele.2018.10.004.

[10] S. S. Yau and J. S. Patel, "A Blockchain-based Testing Approach for Collaborative Software Development".

[11] M. S. Farooq, Z. Kalim, J. N. Qureshi, S. Rasheed, and A. Abid, "A Blockchain-Based Framework for Distributed Agile Software Development," vol. 10, 2022.

[12] K. Duan, J. M. Caballero, and X. Jing, "Scrum-based development model:Improve the engineering quality and testing method of blockchain projects," 2023.

[13] A. Pinna, G. Baralla, M. Marchesi, and R. Tonelli, "Raising Sustainability Awareness in Agile Blockchain-Oriented Software Engineering".

[14] Z. Wu et al., "Kaya: A Testing Framework for Blockchain-based Decentralized Applications".

[15] S. Ji, S. Zhu, P. Zhang, and H. Dong, "Test-Case Generation for Data Flow Testing of Smart Contracts Based on Improved Genetic Algorithm," IEEE TRANSACTIONS ON RELIABILITY, vol. 72, no. 1, 2023.

[16] Y. Murray and D. A. Anisi, "Survey of Formal Verification Methods for Smart Contracts on Blockchain".

[17] C. Benabbou and O. Gurcan, "A Survey of Verification, Validation and Testing Solutions for Smart Contracts," in 2021 Third International Conference on Blockchain Computing and Applications (BCCA), Tartu, Estonia: IEEE, Nov. 2021, pp. 57–64. doi: 10.1109/BCCA53669.2021.9657040.

[18] D. He, R. Wu, X. Li, S. Chan, and M. Guizani, "Detection of Vulnerabilities of Blockchain Smart Contracts," IEEE Internet Things J., vol. 10, no. 14, pp. 12178–12185, Jul. 2023, doi: 10.1109/JIOT.2023.3241544.

[19] M. Barboni, A. Morichetta, and A. Polini, "SuMo: A mutation testing approach and tool for the Ethereum blockchain," Journal of Systems and Software, vol. 193, p. 111445, Nov. 2022, doi: 10.1016/j.jss.2022.111445.

[20] S. Bansod and L. Ragha, "A Quantum Resistant Blockchain System for Privacy Protection of Patient Records," IJETT, vol. 71, no. 4, pp. 79–96, Apr. 2023, doi: 10.14445/22315381/IJETT-V71I4P208.

[21] M. Kuzlu, M. Pipattanasomporn, L. Gurses, and S. Rahman, "Performance Analysis of a Hyperledger Fabric Blockchain Framework: Throughput, Latency and Scalability," in 2019 IEEE International Conference on Blockchain (Blockchain), Atlanta, GA, USA: IEEE, Jul. 2019, pp. 536–540. doi: 10.1109/Blockchain.2019.00003.

[22] S. Kaushik and N. E. Madhoun, "Analysis of Blockchain Security: Classic Attacks, Cybercrime and Penetration Testing," in 2023 Eighth International Conference On Mobile And Secure Services (MobiSecServ), Miami Beach, FL, USA: IEEE, Nov. 2023, pp. 1–6. doi: 10.1109/MobiSecServ58080.2023.10329210.

[23] S. Bansod and L. Ragha, "Challenges in making blockchain privacy compliant for the digital world: some measures," Sādhanā, vol. 47, no. 3, p. 168, Aug. 2022, doi: 10.1007/s12046-022-01931-1.

[24] "TRUFFLE SUITE." ConsenSys Software Inc., 2016. [Online]. Available: https://trufflesuite.com/

[25] ConsenSys Software Inc., "GANACHE." 2016. [Online]. Available: https://trufflesuite.com/ganache/

[26] "REMIX." Ethereum Foundation. [Online]. Available: https://remix-project.org/

[27] "MYTHX." ConsenSys Software Inc. [Online]. Available: https://mythx.io/

[28] "SOLHINT." MIT. [Online]. Available: https://protofire.github.io/solhint/

[29] "HARDHAT." Nomic Foundation / Ehereum Foundation, 2023. [Online]. Available: https://hardhat.org/

[30] "HYPERLEDGER CALIPER." The Linux Foundation - Hyperledger Foundation, 2023. [Online]. Available: https://www.hyperledger.org/projects/caliper

[31] "CUCUMBER." SmartBear Software, 2023. [Online]. Available: https://cucumber.io/