Blockchain based testing approach for Collaborative Software Development

1st Keval Sadharakia
DA-IICT
Gandhinagar, India
201901269@daiict.ac.in

2nd Satyam Bhut
DA-IICT
Gandhinagar, India
201901408@daiict.ac.in

3rd Deep Tank
DA-IICT
Gandhinagar, India
201901410@daiict.ac.in

Mentor: Prof.Jayprakash Lalchandani
DA-IICT
Gandhinagar, India

***Abstract*—It takes a number of teams, including software development teams, domain experts, user representatives, and other project stakeholders, to collaborate in order to develop large-scale and complex software systems. In software development process, testing of the software is one of the most complex and significant process. Because of the many factors like challenges in achieving testing objectives within the allotted time constraints, a lack of effective data sharing policies, ambiguous testing acceptance criteria at various levels of testing, and a lack of reliable coordination among the teams involved in software testing. Information sharing between these teams must be effective, trustworthy, and dependable for software testing to be effective. Software testing methods now in use for collaborative software development make use of centralised or decentralized platforms for communication, knowledge management, and software testing. These approaches have many drawbacks. So we use a new approach that is a blockchain-based testing approach. This paper presents a software testing approach for collaborative software development using a private blockchain and the many different properties of the blockchain.**

***Index Terms*—Private blockchain, collaborative software development, software testing, auditability.**

I. Introduction

The software development process is a complex system involving multiple teams, which require them to work in collaboration with one another. Out of these teams, the software testing team is one them. Software testing techniques involving the usage of private blockchains exist in the current times. It includes the generation of smart contracts with the required criteria like the testing specifications to check the results of the software component accurately.

We have referred to the pre-existing software testing technique mentioned in the paper written by Stephen S. Yau and Jinal S. Patel. They discuss a complete blockchain-based software testing methodology. The process of the generation of smart contracts is also specified in the paper. The method specified in the paper first generated the smart contracts for all the involved teams in the software testing process. Then the test cases are run corresponding to the smart contract they belong to, and the results are then checked with the acceptance criteria mentioned in the smart contract. The flow of the method specified is depicted in fig(1):-

The performance of this method of software testing is also analyzed in the paper. The performance results are compared to the Centralized and decentralized approaches of software techniques. It is shown in the fig(2).

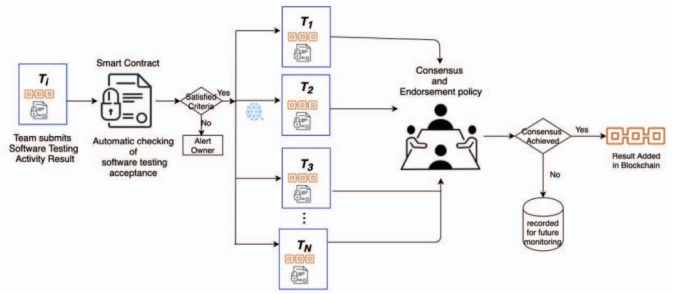

Fig. 1. Flow of the Software Testing Approach

No	Properties	Centralized Approach	Decentralized Approach	Our Approach
i	Immutability	No	No	Yes
ii	Auditability	No	No	Yes
iii	Consensus	No	No	Yes
iv	Non-repudiation	No	No	Yes
v	Private communication	No	No	Yes
vi	Storage Overhead	Low	Moderate	High
vii	Data Processing Speed	High	Moderate	Low
viii	Setup Efficiency	High	Moderate	Low

Fig. 2. Performance Comparison of the Centralized and Decentralized Approach

II. Background Reading

In paper [2], the concepts regarding collaborative software development are given. Collaborative software development consists of a group of people working together to achieve a common goal of developing large-scale and high-quality software. Also, the author discussed the centralized as well

as a decentralized approach to make coordination between various software development teams. A blockchain-based approach has also been developed to solve the limitations of these approaches and acquire features like immutability, auditability, reliability, and trust among the various teams.

Paper [4] aims to identify the challenges encountered and approaches used for blockchain-oriented software testing to ensure high-quality standards. The distributed ledger technology (i.e., blockchain) provides a better platform for a very innovative and secure way to manage transactions. With the properties like safety, tamper-proof, peer-to-peer, and decentralized networks with distributed consensus, blockchain has been named one of the biggest inventions after the internet. Also, a detailed explanation of the smart contract, various metrics of blockchain, and different approaches to blockchain testing have been discussed.

Through the paper [1] the authors Stephen S. Yau and Jinal S. Patel have discussed in detail the testing methods already present and worked on their shortcomings. The methods discussed are the centralized and de-centralized methods of software testing. These methods had their own limitations and hence they proposed the blockchain testing methodology. The paper discusses in depth the formation of smart contracts and then their implementation in the actual software testing process. The steps to be followed in the process of software testing are all discussed. Finally a comparison of the method listed is done on the basis of its performance against the other two pre-existing methods.

III. Motivation

From the fig(2) depicting the performance of the approach, it is evident that traits such as Storage Overload and Data Processing speed are not at all in our favor.

The large overhead storage issue in the approach is because there are multiple copies of the same kind of data being stored in the blockchain, whereas in the Centralised and decentralized approach, there is not much memory requirement. The slow processing speed is because all the transactions are required to go through the consensus criteria before being added to the blockchain and the repetition of similar transactions causes many transactions to go through this audit process by consensus and hence resulting in a very slow processing speed.

IV. Our Approach:

A. *Notations and Process:*

Before moving toward the approach, below are the notations used throughout the paper.

- S: Software Component in the collaborative software development.
- T: Team participating in the development process
- C: The smart contract of an S, which is generated by the management team for testing S.
- R: A testing transaction, which is a transaction involving testing in the collaborative software development project, such as addition of the smart contract of a software component after it is generated by the management team of a software component, and an update of the smart contract of a software component.
- Z: Total number of R's to be generated.
- P: Least number of transactions to be considered while testing the system
- X: Threshold value for each transaction result.
- N: Total number of Possible Transactions
- L: Total number of software components in the project
- T_m: Total participating teams in the development project.

As we have seen that the problem with the first approach is that we are storing each transaction in the blockchain independently of the results. Now to store only eligible transactions that might be of use in the future, we introduce new validity criteria before we decide to store them somewhere.

The validity criteria can be introduced after checking the Smart contract's satisfaction criteria and the consensus criteria. The procedure to validate the transaction is that we define $X\%$ as the threshold validity criteria. The only transactions whose testing results are above this X% will be considered valid transactions and will be stored for future reference, and the transactions failing to do so are assumed to be of no use and dumped.

By introducing this strategy, we are reducing the storage overhead significantly and also increasing the data processing speed, as low data processing speed is due to the consensus protocols used for recording each transaction in the blockchain for data ownership, auditability, and the prevention of injection attacks in the software testing process. Fig(3) represents the flow of our approach:

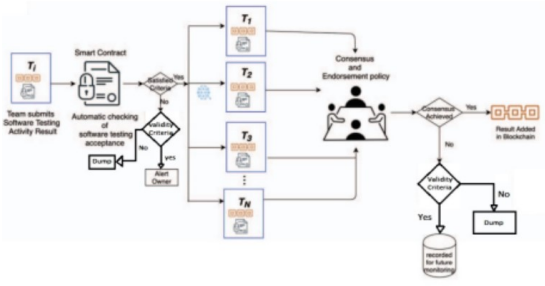

Fig. 3. Flow of the proposed approach

B. *Determining the value X:*

The calculation for the value X is as follows:

For every smart contract C(i,k), Team T_k, produces R_t $(1 \leq t \leq Z)$ results.

We will generate all the possible numbers of transactions for a given smart contract (suppose N). Out of these transactions, we have assumed that the bottom p% of them are considered. Here, the number P will be predetermined by the owner of that smart contract. It'll be already mentioned inside the attributes of the smart contract. The template for the smart contract for this approach is given in fig(4). Now the lowest P% of the N transactions will be analyzed, and the highest fault percentage will be the value of X.

Here, the fault percentage for every transaction will be calculated as AP = $\frac{Received \times 100}{Ideal\_Output\_of\ C(i,k)}$

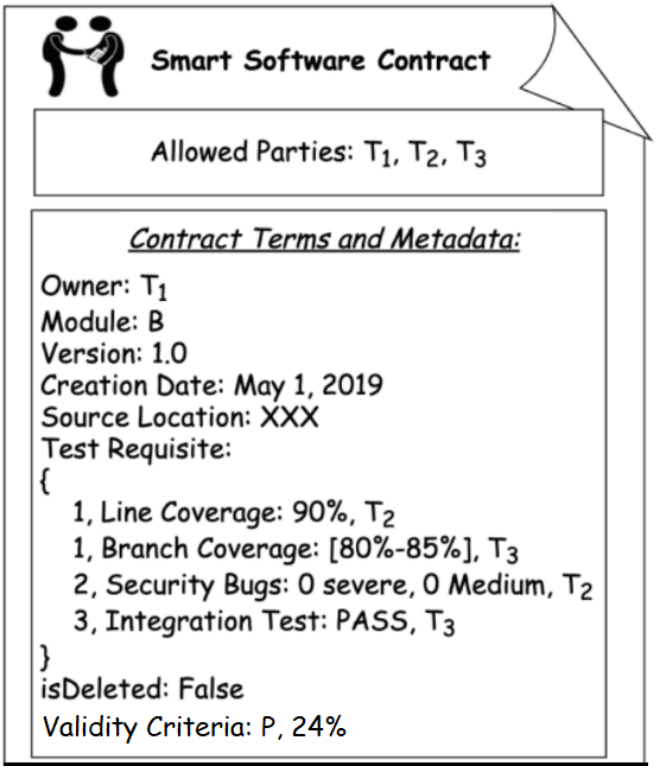

Fig. 4. Template of the smart contract

V. Algorithm and Test Cases

A. *Algorithm:*

1) Management team G generates $C(i,k)_a$, from T_i to T_k, $1 \leq k \leq N$ for a S_a, $1 \leq a \leq L$. Repeat this process for S_a , a = 1, 2, ...L.
2) T_k produces $\frac{N}{4}$ results as per assignment in the $C(i,k)_a$.
3) If the result R_t ,$1 \leq t \leq \frac{N}{4}$, generated by T_k meets the acceptance criteria of $C(i,k)_a$, R_t is added to the blockchain as a blockchain transaction.
4) If R_t generated by T_k does not meet the acceptance criteria of $C(i,k)_a$, T_i is alerted by the blockchain that R_t is not successful, and the fault percentage of the result is as discussed above and stored in the transaction node. Also the value X initialized as 0.
5) The value of P is extracted from the smart contract and the highest Accuracy percentage out of the lowest P% of the results that don't meet the acceptance criteria of the $C(i,k)_a$ will be stored as the value X. and the remaining results will be added to blockchain with the rejection message.
6) All results of the smart contract will be generated and the results whose accuracy percentage falls below the X value will be dumped.
7) Repeat steps 2 to 5 for, $T_K, 1 \leq K \leq T_m$

The activity diagram for the same is as follows:

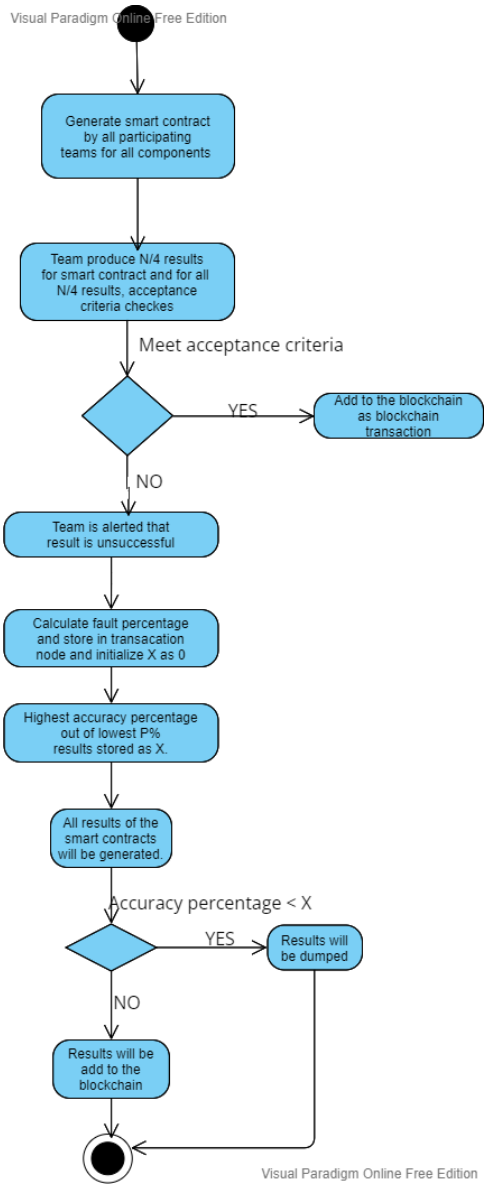

Fig. 5. Flow of the proposed approach

B. *Sample Test Cases:*

Consider the following assumptions for the below test cases. N = 1000, P = 20%

ID	Description	Flow	Expected Result
TC1	Out of the 250 results that'll run, suppose 150 of the results fail to meet the acceptance criteria.	$1-2-3-4-5-6-7$... When the above flow is attained it is determined that the value of X supposedly turns out to be 16%	Out of all the N results, the ones that fail to meet the AC and having AP lower than 16% will be dumped and the rest will be added to the blockchain.
TC2	Out of the 250 results that'll run, suppose all of them pass the acceptance criteria.	$1-2-3$... When the above flow is attained, the value of X will be equal to its initialised value.	All the results will be directly added to the blockchain and none of them will pass validation cycle.
TC3	Out of the 250 results that'll run, suppose all the 250 of the results fail to meet the acceptance criteria.	$1-2-3-4-5-6-7$... When the above flow is attained it is determined that the value of X supposedly turns out to be 18%.	Out of all the N results, the ones that fail to meet the AC and having AP lower than 18% will be dumped and the rest will be added to the blockchain.
TC4	Out of the 250 results that'll run, suppose 1 of the results fail to meet the acceptance criteria.	$1-2-3-4-5-6-7$... When the above flow is attained it is determined that the value of X supposedly turns out to be 27%.(Here the value of X will automatically be the AP of the 1 failed result)	Out of all the M results, the ones that fail to meet the AC and having AP lower than 27% will be dumped and the rest will be added to the blockchain.

TABLE I
TEST CASES

C. *Real World implementation and Test cases:*

In the previous section, we have seen the theoretical test cases and algorithms of our approach. In this section, we will see how our algorithm fits in real-world cases.

Consider the library management system, where the information regarding returning and issuing books is stored in a blockchain. Now the software testing for this system is being carried out as follows:

The owner of the system wants to test the amount to be paid by the reader of the book if he would return the issued book after the mentioned date. Assuming the owner has set the value P = 20% in the smart contract, the expected late amount that would be paid by the reader is 150 Rs per day. Now owner tests the system on 1000 issued transactions, and considering the reader is returning the book after 2 days, then the following cases may occur.

1) Out of the 250 transactions, 100 transactions calculated the late amount as 300 Rs, and from the other 100 transactions, the highest amount that a transaction has attained is 260. Hence the value X, in this case, becomes 87%. So after this, by testing all the transactions, if any resulting amount falls below 87% of the 300 Rs, then that transaction will be discarded, and others will be added to the blockchain.

2) Out of the 250 transactions, all 250 transactions calculated the late amount as 300 Rs. Hence the value X, in this case, becomes 100%. So by testing all the transactions, if any resulting amount falls below 100% of the 300 Rs, then that transaction will be discarded, and others will be added to the blockchain.

VI. CONCLUSION:

In the paper, we have tried improvising the existing collaborative software testing approach. The existing approach already resolved the problems (i.e., immutability, auditability, Non-repudiation, etc.) with the centralized approach. But this approach still faced the problem of storage overhead and low data processing speed. Our approach has improved these problems considerably by introducing more constraints and criteria to the storage space. Hence, in a large-scale real-world system where it will require more transactions to be tested, the given approach will fit quite well and give efficiency and better performance. In nearer future, the setup efficiency can also be improved, and software testing will be done more efficiently and rapidly.

REFERENCES

[1] A Blockchain-based Testing Approach for Collaborative Software Development by Stephen S. Yau and Jinal S. Patel School of Computing, Informatics, and Decision Systems Engineering Arizona State University Tempe, AZ 85287-8809, USA

[2] Generating Trusted Coordination of Collaborative Software Development Using Blockchain by Jinal S. Patel

[3] https://www.investopedia.com/terms/b/blockchain.asp

[4] Blockchain Oriented Software Testing - Challenges and Approaches Rohan Koul Quality Associate Suite Test Engineering SAP Labs India Pvt. Ltd. Gurgaon, Haryana-122002, India