Using Blockchain for Requirements Traceability

Yash Shah
DA-IICT
Gandhinagar, India
201901210@daiict.ac.in

Darshil Shah
DAIICT
Gandhinagar, India
201901232@daiict.ac.in

Jeel Faldu
DAIICT
Gandhinagar,India
201901263@daiict.ac.in

Assigned By : Prof. Jayprakash Lalchandani

Abstract — Requirement Traceability is unavoidable. In day-to-day life, we do tracing everywhere when we search for information, and it becomes difficult to imagine a software development environment without tracing. In this article, we propose and develop a blockchain-based architecture to enhance requirement traceability with the help of Solidity. The main motive for using Blockchain technology is its Immutability feature which provides complete security and protection to data inside the blocks. Through this blockchain-based traceability system, we can store, record, and trace the requirements efficiently.

Nowadays, so much information is needed and created during software creation; requirements tracing is crucial. Designing complex systems can be challenging to recall all the connections built to link the data. In a distributed or multi-team development, it may even be impossible to be aware of such relationships. Furthermore, the situation becomes more complicated when direct ties to a specific piece of information are not necessary. Usually, one wishes to track a series of connections that extends past the trace's initial point. Requirement Traceability is a crucial document in the field of software engineering which maps the requirements with the test cases. It is one of the most critical steps in the Software development life cycle. RTM is a mechanism that traces out the requirements with its corresponding test cases. The primary purpose of RTM is to validate that all requirements are checked via test cases such that no functionality is unchecked during Software testing. The requirements associated with software development include business requirements, user requirements, UI requirements, functional and non-functional requirements, and technical requirements. RTM and the test cases also show the status of corresponding test cases, i.e., they have been passed or failed. This would help the testing team understand the testing activities for the specific products.

RTM is a powerful planning tool for determining the number of tests required, the types of tests required, and whether these can be automated, done manually, or whether any existing tests can be reused. Requirements traceability helps minimize error, maximize the success of product development, and develop higher quality products. It helps to smoothen the process of product development by providing easy analysis of the change in the product, for verification and validation of the product, etc.

Requirement Traceability is classified into forward traceability, backward traceability and bidirectional traceability.

Forward Traceability:- All types of requirements proposed by the clients are mapped with a test case so that all the requirements are tested effectively. We can add extra test cases, i.e., expanding the scope of the project.

Backward Traceability:- Mapping of test cases with corresponding requirements is carried out. So we cannot go beyond our requirements, i.e., we can not expand the scope of the project by adding some extra code or design elements which is not mentioned in the Requirements.

Bidirectional Traceability:- A single document that is an amalgamation of forward traceability and backward traceability. The primary purpose of this type is all the test cases are specified with their corresponding requirements.

The advantages of using RTM include 100% test coverage, highlighting any missing requirements or document inconsistencies, and focusing on each test case's execution status and its requirements.

Requirement traceability matrix can:

1. Show the required coverage in the number of test cases.
2. Design status as well as execution status for the specific test case.
3. If there is any testing method, i.e., User Acceptance test, integration testing, etc to be done by the users, then the status of each and every testing method can also be captured in the same matrix.
4. The related defects and the current state can also be mentioned in the same matrix.
5. User stories corresponding to the requirements are also captured.
6. Test designer, developers, and user stories designer-related information can also be mentioned.
7. Designing (UI) related aspects can also be included in the matrix.

The fields which we have included in our matrix are as follows :-

1. Requirement ID
2. Requirement description
3. User stories
4. Test designer
5. Test Case ID

6. Status
7. Unit testing
8. System testing
9. Integration testing.

Blockchain :-

Bitcoin is one of the most widely used applications of blockchain technology. It has been defined as distributed digital ledgers that keep records and transactions as encrypted time-stamped chains. There is no involvement of third parties while performing any transaction. Blockchain technology is a peer-to-peer network and hence the validation and verification of transactions are done by network participants.

Key terminologies of blockchain include distributed data storage, consensus mechanisms, time-stamp, encryption algorithms, etc. The time-stamp function is used to record the history of transactions and helps in identifying the correctness of transactions. Each transaction is stored in each block and forms a chain of blocks termed Blockchain. One cannot modify or remove information without network verification. This feature, immutability, enhances the trustworthiness of information, greatly reducing the probability of information falsification, fraud, or corruption.

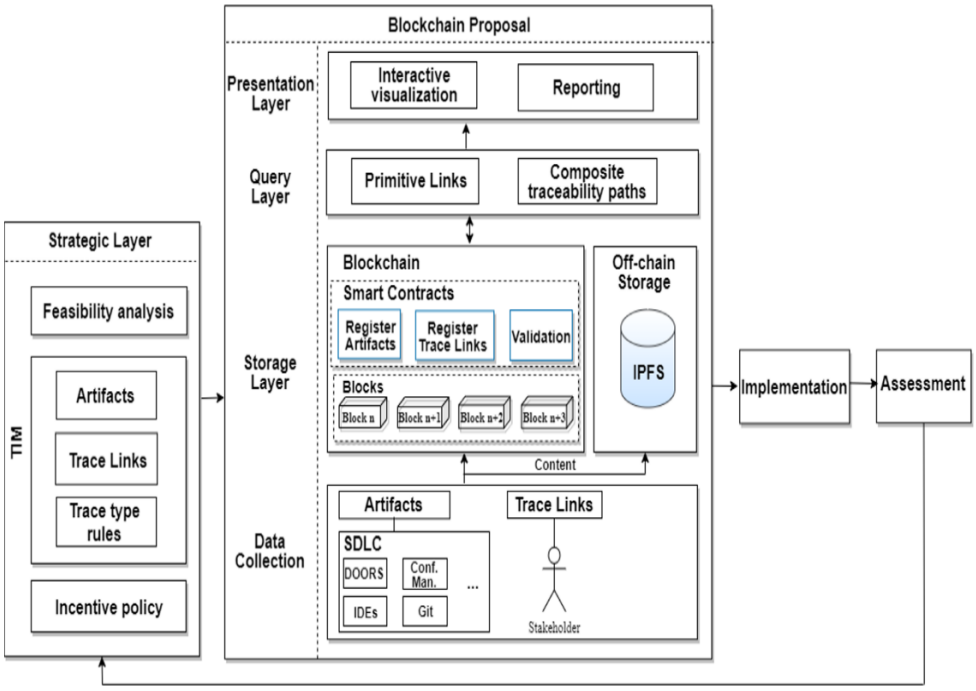
A smart Contract is a computer program that automatically performs the transactions between the parties when certain conditions are met. Smart Contract is involved in blockchain, making transparent, inexpensive, and immutable. Transactions are performed speedily as tasks are performed automatically, so no human interaction takes place. They eliminate the intermediates in the process, making it cost-effective. All the above-mentioned features of the Blockchain can effectively protect data by ensuring consistency and reliability. Blockchain technology is decentralized i.e., managed and executed by all nodes in the network; hence, there is no dominance of a single party.

Example: Traceability Of Original Achievements.

Initially, registration of original documents takes place. After the creation of document rights, the document gets followed. The creator sends all document relevant information to the blockchain via front-end application interface. A block is created, and all information gets embedded inside it. Now the patent agency will check whether any similar document exists in blockchain. If yes, then the registration process will be stopped and that block would not be added in the blockchain. If no, all further steps are performed, and that block will be added to the blockchain. So this technology helps us to protect and preserve the original data and prevent it from modification.

Proposed Framework :-

Blockchain proposal forms the core component of our model and it consists of Data collection, storage layer, query layer and presentation layer. Data collection includes gathering information regarding requirements, test cases and designers. All this information gets stored inside a block and it gets added inside the blockchain. Smart contracts are used to store requirements, trace, validate their trace and reward trace link creators etc.



Query layer also plays an important role in storing the information. Source code of any software resides in this region.

Presentation layer is responsible for visualizing the traceability related information. Graphical and tree approaches can be used to visualize at small scale and hierarchical approach at large scale to understand in a better way.

Implementation :-

In this part we have developed and implemented a proposal related to blockchain technology. Therefore cooperation is required between blockchain developers and requirement traceability experts. Mapping of requirements with artifacts is done efficiently using blockchain. We have implemented functions which can modify the status of test cases and different testing methods. We can also get all information corresponding to a particular requirement. We can directly get the status of test cases in constant time using mapping data structure. One should make sure that all participants that are involved in the software development life cycle work transparently and efficiently to overcome the resistance. This blockchain based framework will also motivate the stakeholders to participate in creation of Requirement Traceability Matrix.

```
pragma solidity ^0.5.1;
import "@nomiclabs/buidler/console.sol";
pragma experimental ABIEncoderV2;

contract MyContract {
    RTM[] public Requirements; // RTM Matrix
    uint public myint = 0; // assign testcase ID
    uint256 public ReqCount=0; // Total no of Requirements for any specific
system

    struct RTM {
        uint ReqNo;  // Requirement No
        string ReqDes;   // Description of the requirement.
        string userstories; // userstory corresponding to requirement
        string test_designer;  // information about person who designed test
case
        uint[] TcId;       // Testcases for that specific Requirements
        uint[] status;      // status of TestCases i.e. either 1/0.
        uint[] unit_testing;  // whether unit testing is required or not for
any testcase
        uint[] system_testing; // whether system testing is required or not for
any testcase
        uint[] integration_testing; // whether integration testing is required
or not for any testcase
    }

    mapping(uint => uint[]) private map;   // forward Traceability by mapping
```

```
of Requirements with Testcases
    mapping(uint => uint) private map1;  // mapping of Testcases with their
status
    mapping(uint => uint) private unit_map; // displays status of unit testing
for any testcase
    mapping(uint => uint) private system_map; // displays status of system
testing for any testcase
    mapping(uint => uint) private integration_map; // displays status of
integration testing for any testcase
    mapping(uint => string) private testcase_map; // mapping of testcase with
their description

    // function to add any requirements by providing sufficient information

    function addReq(uint _ReqNo,string memory _ReqDes,string memory
_userstories,string memory person) public {
        uint[] memory newarray = new uint[](4);

        newarray[0] = 0 + myint;
        newarray[1] = 1 + myint;    // Testcase ID for the requirements
        newarray[2] = 2 + myint;
        newarray[3] = 3 + myint;

        uint[] memory newarray1 = new uint[](4);

        newarray1[0] = 1;
        newarray1[1] = 0;
        newarray1[2] = 1;       // boolean values denoting status of test
cases.
        newarray1[3] = 0;

        map1[newarray[0]] = newarray1[0];
        map1[newarray[1]] = newarray1[1];
        map1[newarray[2]] = newarray1[2];   // mapping testcases with their
status
        map1[newarray[3]] = newarray1[3];

        mapping_unit(newarray,newarray1);
        mapping_system(newarray,newarray1);
        mapping_integration(newarray,newarray1);
        mapping_testcase(newarray,person);


Requirements.push(RTM(_ReqNo,_ReqDes,_userstories,person,newarray,newarray1,new
array1,newarray1,newarray1));  // storing all above information inside the
```

```
matrix
        map[_ReqNo] = newarray;  // mapping of requirements with RTM
        ReqCount += 1;
        myint += 4;
    }

    // mapping_unit function is used to store the values inside mapping
("unit_map") data structure

    function mapping_unit(uint[] memory arr,uint[] memory arr1) public
    {
        for(uint i=0;i<4;i++)
            unit_map[arr[i]] = arr1[i];
    }

    // mapping_unit function is used to store the values inside mapping
("system_map") data structure

    function mapping_system(uint[] memory arr,uint[] memory arr1) public
    {
        for(uint i=0;i<4;i++)
            system_map[arr[i]] = arr1[i];
    }

    // mapping_unit function is used to store the values inside mapping
("integration_map") data structure

    function mapping_integration(uint[] memory arr,uint[] memory arr1) public
    {
        for(uint i=0;i<4;i++)
            integration_map[arr[i]] = arr1[i];
    }

    // mapping_unit function is used to store the values inside mapping
("unit_map") data structure

    function mapping_testcase(uint[] memory arr,string memory _val) public
    {
        for(uint i=0;i<4;i++)
            testcase_map[arr[i]] = _val;
    }

    // function returns all information for that particular Req No.

    function get(uint no) public view returns(RTM memory) {
```

```
        return Requirements[no-1];
    }

    // BLOCKCHAIN IMMUTABLE PROPERTY is enhanced using below function

    // function changestatus(uint256 no, string memory _val) public view  {
    //     map1[no] = 5;
    // }

    function getTCfromReqNo(uint no) public view returns(uint[] memory) {
        return map[no];
    }

    // Below mentioned functions are used to change the value of unit testing,
system testing, integration testing and
    // status if in future some changes have occured in software.

    function change_unit_testing(uint no,uint[] memory arr) public {
        Requirements[no-1].unit_testing = arr;
        mapping_unit(Requirements[no-1].TcId,arr);
    }

    function change_system_testing(uint no,uint[] memory arr) public {
        Requirements[no-1].system_testing = arr;
        mapping_unit(Requirements[no-1].TcId,arr);
    }

    function change_integration_testing(uint no,uint[] memory arr) public {
        Requirements[no-1].integration_testing = arr;
        mapping_unit(Requirements[no-1].TcId,arr);
    }

    // Whether the testcase is successfully implemented or not is displayed
using "getstatusofTC" function

    function getstatusofTC(uint _val) public view returns(uint) {
        console.log("The status of testcase" , _val ," is" , map1[_val]);
        return map1[_val];
    }


}
```

Related Work :-

Huang proposed a model which helps stakeholders to execute their traceability methods using a graphical approach. It consists of four layers: strategic, document management, query and executable layer. Strategic layer is used to store the artifacts and trace paths. Document management layer lists out all sets of artifacts. Query layer builds a query between types of artifacts and their traceability path. Executable layer provides the user interface and helps in visualizing the result of queries. We have also used the idea of layers but simulated them in different manners.

Elamin and Osman proposed a user defined model to store artifacts and their trace but their representation and storage were their problems. In order to overcome this, they used the concept of graphs. With the help of graphs they used to attain all necessary functionalities but main limitations was it was centralized and hence was not distributed.

So nowadays we see that blockchain technology is playing a crucial role in software engineering. There were other studies also which tried to improve the integrity of the software development process. All this studies are good start using blockchain technology in software engineering but none of them were used in distributed systems.

Conclusion :-

In this paper we proposed a blockchain based framework for requirement traceability. Blockchain technology is used as the backbone of SDLC. What are the requirements, time of creation, information about designer, various testing approaches, status of test case etc all this information is present in block. It allows us to get information of requirements and we can change the status of the test case if that functionality is achieved in future.

Blockchain is one of the best way to implement it as it ensures data protection, security and immutability. It ease the software development life cycle and performance of the software is improved. It also motivates the stakeholders to put their requirements and their corresponding test cases using blockchain.

There are also some limitations for this model. Our model is not distributed and so the scalability of the system is poor. Manual creation of the smart contract and validation of trace links are also some issues related to it.

References :-

1. Murugappan, S., Prabha, D.: Requirement traceability for software development lifecycle. Int. J. Sci. Eng. Res. 8, 1–11 (2017).

2. Elamin, R., Osman, R.: Implementing traceability repositories as graph databases for software quality improvement. In: 2018 IEEE International Conference on Software

Quality, Reliability and Security (QRS), pp. 269–276 (2018)

3. Mader, P., Gotel, O., Philippow, I.: Getting back to basics: promoting the use of a traceability information model in practice. In: 2009 ICSE Workshop on Traceability in Emerging Forms of Software Engineering, pp. 21–25 (2009)

4. Cleland-Huang, J., Berenbach, B., Clark, S., et al.: Best practices for automated traceability. Computer 40, 27–35 (2007).

5. Farshidi, S., Jansen, S., España, S., Verkleij, J.: Decision support for blockchain platform selection: three industry case studies. IEEE Trans. Eng. Manag. 67, 1109–1128 (2020)

6. Peng Jhu, Jian Hu, Xiaotone Li, Qingyun Zhu. Using Blockchain Technology to Enhance the Traceability of Original Achievements. IEEE , 15 April 2021.