Blockchain Frontier Technology (B-Front) **P-ISSN: 2808-0831**
Vol. 1 No. 2 January 2022 **E-ISSN: 2808-0009**

An Agile Software Engineering Method to Design Blockchain Applications

Ita Erliyani[1], Siti 'Afifah Wulandari[2]
University of Raharja, Indonesia
Jl. Jenderal Sudirman No.40, RT.002/RW.006, Cikokol, Kec. Tangerang, Kota Tangerang, Banten 15117[1,2]
e-mail: ita.erliyani@raharja.info[1], siti.afifah@raharja.info[2]

Erliyani, I., Wulandari, A., An Agile Software Engineering Method to Design Blockchain Applications
Blockchain Frontier Technology (B-Front), *1*(2), 84-95.
DOI : https://journal.pandawan.id/b-front/article/view/52

Author Notification
01 January 2022
Final Revised
01 January 2022
Published
01 January 2022

Abstract

Cryptocurrencies and its underlying technology, Blockchain, are transforming banking and economics by allowing trustworthy programs to exist without the requirement for a trusted counterpart. In recent years, the Blockchain and the programs that run on it, known as Smart Contracts, have seen increased use across all businesses that demand trust and verifiable credentials. Some have compared the "Blockchain revolution" to the early days of the Internet and the World Wide Web. As a result, all Blockchain-based software development is accelerating at a dizzying pace. Many software engineers argue that the widespread interest in Blockchain technology has resulted in haphazard and rushed software creation, a type of first-come, first-served competition that fails to assure software quality or take into consideration core software engineering ideas. This research tries to solve this issue by proposing a software development strategy for gathering requirements, assessing them, designing, creating, testing, and deploying Blockchain applications. Several Agile approaches are employed in the process, including User Stories and iterative and incremental development based on them. It does, however, employ more formal notations, such as UML diagrams that define the system's design, with improvements to express unique Blockchain concepts. The method is well discussed, including an example to illustrate how it works.

Keywords*: Blockchain, Smart Contracts, Software, Agile*

1. Introduction

The so-called Blockchain applications are currently one of the most popular IT topics [1]. The Blockchain is a decentralized and secure technology that was created to manage the Bitcoin money [2]. Developers immediately discovered that the Blockchain could also be utilized as a decentralized computer, running Smart Contracts - programs that serve as the foundation for automated contract enforcement. The realization of the technology's promise - the ability to enforce contracts, eliminate middlemen, and overcome space and time limits – sparked a surge of interest in Blockchain applications [3]. Some analysts believe that "we should conceive of the Blockchain as another class of item like the Internet" and that "broad adoption of Blockchain technology has the ability to reshape the present financial services technical infrastructure."
As a result of this enthusiasm, a growing amount of money is being invested in Blockchain projects [4]. Despite recent market shrinkage, the market capitalization of cryptocurrencies is well over 200 billion US dollars, and venture capital investments, both from traditional funds and

from recent Initial Coin Offerings (ICOs), have surpassed 10 billion US dollars in the last year. This rush to engage in new ventures, usually by swiftly building apps to be the first on the market, has resulted in several major disasters, owing to poor software design and security procedures.Attacks on the so-called "Exchanges," websites where cryptocurrencies may be traded against each other or even against fiat currencies like the US dollar and euro, are common, resulting in alleged losses of well over a billion dollars [5]. Many software developers have expressed their dissatisfaction with the widespread interest in Blockchain technology, particularly the numerous software projects that have sprang up and evolved swiftly around various Blockchain implementations or applications. The scenario is one of a first-come, first-served competition that does not ensure software quality or that the fundamental notions of software engineering are taken into account [6].

The initial phase in fostering a product framework using strong computer programming rehearses is to have a characterized improvement process set up, just as configuration approaches and documentations that are proper for the job needing to be done. Explicit turn of events, test, arrangement, and security evaluation procedures can be applied accordingly. The reason for this article is to propose and test a plan and improvement approach for Shrewd Agreement based Blockchain applications [7]. The whole cycle is generally founded on the Nimble Statement's ideas, with a few extra documentation and practices. The proposed method depends on the way that a Savvy Agreement is a product program that sudden spikes in demand for all hubs of a Blockchain and should have similar results and state on all hubs. Accordingly, a SC isn't permitted to contact the rest of the world in any capacity - it can react to requests through a public connection point and send solicitations to different SCs on a similar Blockchain. Accordingly, the proposed technique parts the Blockchain programming framework definition into two sections: the particular and advancement of the SCs, and the plan and improvement of the product application(s) that cooperate with outer clients and the SCs [8]. To represent Blockchain specificities, the recommended strategy additionally incorporates a language structure that consolidates the UML Use Case, Grouping, and Class outlines.
The proposed method has been scrutinized in various certifiable drives at our college and a side project organization [9].

The remainder of this paper is spread out as follows. We show important work in something very similar or comparable spaces in Segment 2. Segment 3 clarifies the proposed methodology just as the progressions made to a few UML charts to oblige SC thoughts. Segment 4 is an improved model dependent on a genuine situation. Segment 5 finishes up with discoveries and ideas for future examination [10].

2. Related Work

The subject of configuration approaches and, all the more extensively, programming rehearses equipped towards the making of Blockchain-Situated Programming (BOS) is as yet in its outset. The principal call for BOS Designing, who push for the exploration and execution of incredible designing cycles to guarantee fruitful testing, further develop collaboration in huge groups, and make Savvy Agreement creation simpler. They likewise guarantee that current plan documentations may be changed to more readily communicate BOS show a Blockchain scientific classification [11].

Propose a flowchart and an early agenda to help plan choices dependent on Blockchain highlights. offer a strategy for figuring out what parts of an application configuration could profit from Blockchain innovation [12]. They recognize members, their trust connections, and collaborations to make a design that joins Blockchain innovation into current programming frameworks or makes new frameworks that exclusively use Blockchain specifically regions. To offer a procedure for the improvement of Blockchain use cases, utilize an activity configuration research strategy and situational technique designing [13]. They put the idea under serious scrutiny in four unique businesses: banking, protection, development, and cars [14].

A few articles have been distributed proposing enhancements to the Brought together Displaying Language to all the more likely express different fields to manage BOS. Baumeister and his partners Proposed a Hypermedia UML expansion, which included new generalizations and another Navigational Design Model [15]. Moreover, Baresi et al. join underlying and

navigational deliberations provided by hypermedia web models with practical and conduct natives given by UML in a solitary system. Rocha and Ducasse have all the more as of late shown three free demonstrating procedures dependent on notable computer programming models – E-R graphs, UML, and BPMN – and applied them to a Savvy Agreement plan model. They propose specific changes to the UML Class Chart to appropriately communicate Savvy Agreement standards [16].

3. Method

The suggested BOS design process is carried out in a series of phases, which are described in Fig. 1 as a UML activity diagram.

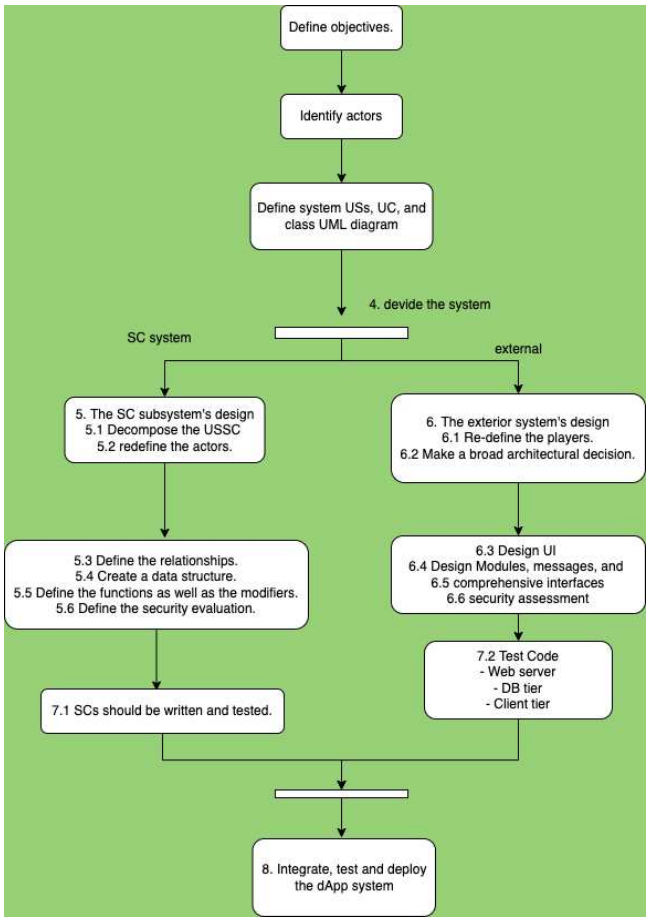

Figure 1. The steps of the proposed BOS development process.

In deeper detail, the proposed BOS development process is the following:

1. Determine which players (human roles and external systems/devices) interact with the system. You may use the concept of evaluating actor trust/untrust to determine if a Blockchain system is truly needed, and for what purposes [6].
2. Create user stories or features to represent the system needs. During this stage, the entire system to be constructed should be considered. It makes no difference if it is built on a Blockchain or on a group of cloud servers.
3. Split the system into two parts:
   4.1 The Smart Contracts, which make up the Blockchain system.
   4.2 An external system that communicates with the first, transmitting transactions to the Blockchain and receiving results.

4. The SC subsystem's design:
   5.1 Redefine the actors and user stories, as in stages 2 and 3, but only for those that interface directly with the SC subsystem and any external SCs that may be utilized;
   5.2 Specify the decomposition in SCs (one or more); define the libraries and external SCs that will be utilized; design the inheritance structure and interface use;
   5.3 Define the message and Ether transfer links and flow, as well as the state diagram (if applicable).
   5.4 Define the data structure, the API, and the events.
   5.5 Define the modifiers and internal functions.
   5.6 Define the tests and procedures for security assessment.
5. The external subsystem's design:
   6.1 Redefine the actors and user stories as in stages 2 and 3, but with the addition of the new (passive) actors represented by the SC system; design the subsystem's acceptance tests.
   6.2 Determine the system's overall design, taking into account the server and client applications, as well as the Blockchain node(s) to be used;
   6.3 Define the user interfaces for all essential modules, including applications.
   6.4 Conduct a system analysis, defining the system's decomposition into modules, message flow, the structure and storage of permanent data, including those anchored to the Blockchain via hash digest memorization, and the data or class structure of the application(s); the connections and data flows between participants, including the SCs, must comply with the analysis of step 5.3;
   6.5 Define the state diagrams (if necessary), comprehensive interfaces of the various modules, and the reaction to SC events.
   6.6 Conduct an external system security evaluation.
6. Develop and test the systems; concurrently:
   7.1 Begin by writing and testing the SCs' data structure and functionalities;
   7.2 Use an agile strategy (Scrum or Kanban) to implement the USs of external subsystems.
7. Complete the total system's integration, testing, and deployment.

3.1 UML diagrams for Smart Contracts

The Strength programming language is normally used to make SCs for Ethereum. Robustness is an article situated programming language, where agreements are determined as classes, with an information construction, public and private strategies, and the capacity to acquire from different agreements. SCs have their own arrangement of thoughts, like occasions and modifiers. We use UML graphs to help with the displaying of SCs. Nonetheless, in light of the fact that SCs have specific novel properties, we added a few extra plans to these graphs to appropriately portray and indicate them [17].

These ideas are presented as UML generalizations, which are labels that might be used in UML graphs any place they are required, at whatever point practicable. In a couple of different conditions, for example, the exchange of Ethers in grouping charts, we needed to utilize an extraordinary documentation. We utilize the accompanying UML charts to display SCs: Class diagrams, which depict the structure and relationships of SCs; in this form of diagram, we added several stereotypes, Statecharts, which are used to depict the different states of a SC and do not require any additional concepts, Sequence diagrams, which depict messages transmitted to a SC and from a SC to another SC; this diagram requires additional types of communications, such as Ether transfers [18].

To portray SCs and structs, UML class outlines are used. In spite of the fact that classes are absent in Robustness, SCs are amazingly tantamount to them. A SC can contain an information design, public and private techniques, and can acquire from at least one SCs, similar as a class. SCs, then again, are extraordinary in that they are shaped by exchanges, however every exchange can make one SC. SCs, then again, can speak with different SCs on a similar Blockchain. In Strength, structs, or modern information structures without capacities, can likewise be characterized. Accordingly, the model of an exchange's SC may

include different SCs it gets from, the used structs, and the outside SCs to which messages are conveyed [19]. Other Ethereum SC-explicit ideas incorporate occasions, which are banners that are raised when something significant happens and sign it to the rest of the world (which should independently notice the SC and act likewise), and modifiers, which are exceptional capacities that are called before a capacity, really taking a look at its imperatives and conceivably halting the execution. Table 1 exhibits the generalizations we set up to permit SC thoughts to be addressed in UML class charts. Beside the compartments holding the name, characteristic, and capacities, the occasions may be addressed in another compartment (activities).

Table 1. shows the changes to the UML class diagram (stereotypes).

Stereotype	Position	Description
«contract»	Class symbol – upper compartment	Denotes a SC.
«library contract»	same as above	A contract retrieved from a (common) library
«struct»	same as above	A struct that holds data but does not perform any operations and is specified and used in the data structure of a contract.
«enum»	same as above	An enum is a type that stores just a list of potential values.
«interface»	same as above	Only function declarations are included in this contract.
«modifier»	Class symbol – lower compartment	Solidity defines a certain type of function.
«array»	Role of an association	An array is used to implement the 1:n relationship.
«map»	same as above	A mapping is used to implement the 1:n connection.
«map[uint]»	same as above	A mapping from integer to value is used to implement the 1:n relationship.

The going with three models portray how one relationship is executed in a SC's data structure [20]. The group and the arranging are the two maintained collections in Heartiness for regulating accumulating (data everlastingly kept on the Blockchain). The past is a standard chief bunch, all things considered, with the extra component of being expandable (yet not killed). The arranging is a variety that can hold key-regard consolidates and get a value promptly given its key, at this point it can't rehash over its parts. The last speculation associates with a normal Vigor programming instance of involving a preparation with positive numbers as keys so it may be iterated over [21]. To address messages, UML Progression Charts are utilized. Messages in Ethereum are associated with trades submitted to the Blockchain by external customers, systems, or SCs.

Messages are indistinguishable from "public limit calls" in object-arranged vocabulary.The matching message can be passed on for nothing expecting that a limit doesn't make or change the Blockchain (it's known as a "view" work). Various correspondences ought to be paid in GAS before they may be sent. The sorts of individuals (portrayed by their records) and such messages are the intriguing components of Ethereum illuminating. Table 2 shows how predispositions may be used to restrict the individuals. A special documentation is required for the correspondences [15].

Stereotype	Description
«person»	A human role that involves utilizing a wallet or other program to transmit messages.
«system»	A third-party system that can deliver messages to the Blockchain.
«device»	A gadget that can deliver messages (usually IoT).
«contract»	A SC that is either internal to the system or external to it.
«oracle»	A type of SC in which the dates are written by a trustworthy third party and which allows access to information about the outside world.
«account»	An Ethereum wallet that only stores Ethers. If the owner activates the transfer, it can only accept Ethers or send Ethers to another account or SC.

Table 2. UML Sequence Diagram Additions (stereotypes).

The following are the many types of communications that are significant to the design:

1. SC creation: it is transmitted by an external participant or another SC; a creation is depicted in a sequence diagram by drawing the new participant at the time level of its formation. Function call: this is the typical "synchronous" or "asynchronous" message that involves Blockchain change and consequently GAS payment.
2. View/pure function call: a transaction with no Blockchain change and no GAS payment; it may be represented by prefixing the message name with the words «view» or «pure».
3. ETH transfers: a transaction in which Ethers are transferred from one account or SC to another. A specific arrow, akin to the inheritance arrow in UML class diagrams, is used to model this.

3.2 AN EXAMPLE OF APPLICATION

An illustration of a product program The referenced SC improvement method is utilized by our College bunch just as the organizations we help. Among the activities being created are a store network the executives framework, a framework for overseeing impermanent work contracts, different remote democratic frameworks for nearby states, and an organization's Investors' Gathering and governing body gatherings. We give a worked on adaptation of the democratic framework to act as an illustration of the underlying periods of the prescribed approach.The first stage is to figure out what the framework's point is. The executives of remote democratic in corporate congregations, including the check of legitimate numbers and the appointment of intermediary votes. The subsequent stage is to recruit entertainers. In the framework, there are basically two entertainers: The corporate chairman is responsible for the framework, just as controlling the investors and their portions, meeting gatherings, and calling for casting a ballot. Goes to gatherings, projects casts a ballot, and has assigned

gathering support to another investor. The third stage is to make client stories. Fig. 3 shows the players and the USs they are engaged with utilizing an UML Use Case outline, where the utilization cases are, indeed, USs. It's significant underlining that these

Americans just notice the democratic interaction, not the innovation that was used to do it. Regardless of whether the execution utilize a Blockchain, they'd be correct.

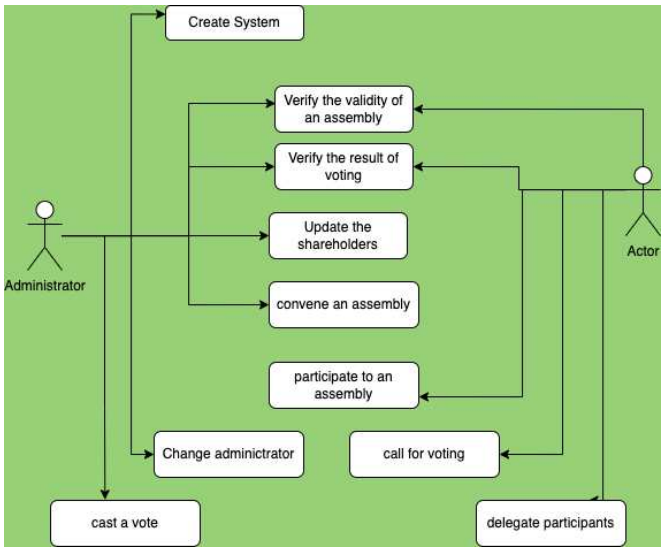

Figure 2. The system specification's User Stories.

We don't have enough area to depict the Americans in depth here. Rather, we exhibit the UML class diagram obtained from an analysis of the supplied USs in Fig. 4. Again, this diagram is not tied to a specific voting system implementation; it simply depicts the entities, data structures, and activities that emerge from the USs of Fig. 3. Step 4: Split the system into two halves. Because all USs employ Smart Contracts, the subdivision is simple in this scenario. The external app subsystem's USs are same. Each one adds the Blockchain as a new player.

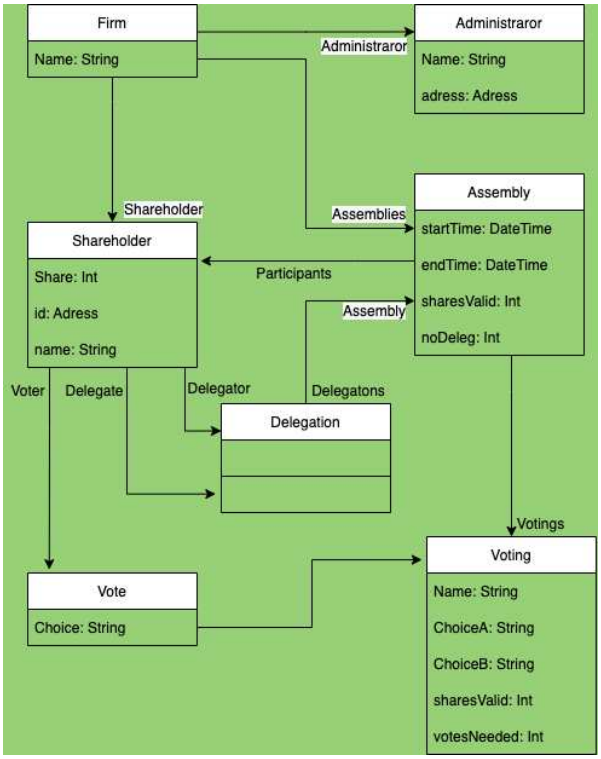

Figure 3. The UML class diagram created from the USs

The UML class diagram created from the USs is shown in Figure 3. The Blockchain subsystem's USs are same. The Actors' IDs are their unique addresses: Corporate administrator: her or his address is first the address used to form the contract, and subsequently it may be changed by the Change administrator US. Shareholders: the Corporate Administrator specifies and manages their addresses. Step 5: Create the SC subsystem design. Because the SC system is so basic, a single SC appears to be the best solution. The "Ownable" standard abstract contract is used to maintain the Administrator's ownership on the SC, following a well-known standard:

```
contract Ownable {
 address public owner;
 modifier onlyOwner() {
 require(msg.sender == owner);
 _;
 }
 }
```

Table 3 shows the results of this function. Other modifiers are added to Table 3 at the bottom to accommodate for other limitations that apply to several functions.

Modifier	Action - Notes
onlyOwner()	Ensures that the message's sender is the contract's owner (the Administrator). Ownable's basic contract inherited it.
onlyShareholder()	Ensures that the message's sender is one of the contract's registered shareholders.

Only Owner or Shareholder()	Ensures that the message's sender is either the contract's owner or one of the shareholders.
Assembly Running()	Asserts that an assembly is truly executing at the moment of the call.
Assembly Not Running()	Assures that no assembly is in progress at the time of the call.

Finally, the contract's public functions put the User Stories' features into action. The findings are summarized in Table 4. We give each function a name (perhaps followed by the function's kind, in this case "view"), call restrictions, modifiers, parameters, and a description of its purpose. In this example, the constructor is the only function that can create a contract. In more intricate cases, a contract may result in another contract, however this is not common. The UML sequence diagrams representing the interactions between Actors and SCs in this example are not representational. All communication calls are exchanged between the SC and an Actor in reality (Administrator or Stakeholder). In this basic scenario, there are no direct connections between more than two persons. As a result, no sequence diagrams are provided.

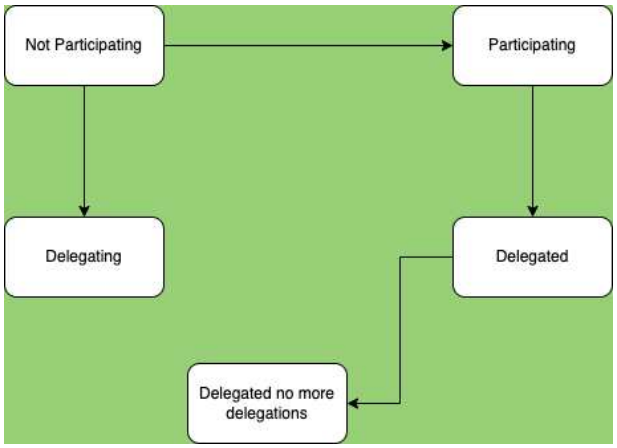

Figure 4. The statechart UML diagram depicts the state of a stakeholder who is attending an assembly or delegating to another stakeholder.

Function	Modifiers Parameters	Action - Notes
constructor	string nameFirm string nameAdmin [(string nameSh, address addrSh, uint16 noShares)]	Create the VotingManagement contract by entering the firm's name, the Administrator's name, and the name, address, and number of shares for each shareholder.
addShareholder	onlyOwner string nameSh address addrSh uint16 noShares	Fill in the details for a new shareholder, including his name, address, and number of shares.
Delete Shareholder	onlyOwner address addrSh Delete the given shareholder, giving his address.	Only if the shareholder does not actively participate in an assembly can this be done.

editShareholder	onlyOwner address addrSh string nameSh uint16 noShares	Update the specified shareholder's information, including his (unchangeable) address, name, and number of shares. Only if the shareholder does not actively participate in an assembly can this be done.
Change Administrator	onlyOwner address newOwner string nameAdmin	Give the new administrator's name and address.
Convene Assembly	onlyOwner	Convene an assembly, specifying the start and finish dates and times, a brief description, the minimum percentage of shares required for validity, and the maximum number of delegates that a single Shareholder can get. There can be no overlap between the old and new assemblies.
addVoting	onlyOwner	Add a voting call to the specified assembly, defining the name of the vote, the two alternatives to choose from, the minimum percentage of voting shares, and the number of votes required for a valid vote. It's unlikely that the assembly has already begun.

Assuming that you follow the techniques above, composing the SC is clear. Besides, it doesn't include Ether moves, beside the GAS important to finish the exchanges, and its security issues are unimportant attributable to its effortlessness and simplicity of actually looking at the preconditions of its messages. With the end goal of straightforwardness, we will avoid the plan and execution of the outer framework (stage 6). It incorporated the advancement of a responsive application that put away the Entertainers' location and private key, permitting them to send messages to the SC, which ran on the Ethereum Blockchain. This application was fabricated utilizing node.js and web3.js on the customer side. To interact with the Ethereum Blockchain, the Ethereum Javascript library Web3.js is used.

4. Conclusion

In spite of the enormous exertion by and by put into creating DApps, helpless computer programming rehearses are as yet being utilized in BOS programming advancement. Truly, researchers are as yet chipping away at apparatuses or strategies for demonstrating and controlling the eccentricities that a product engineer should manage while managing Blockchain Situated programming frameworks. To match this new programming worldview, customary programming devices and cycles should be refreshed and adjusted.

By furnishing engineers with instruments like those utilized in conventional computer programming to address structural plan, security concerns, testing planes and techniques, and further develop programming quality and upkeep, a sound computer programming approach could incredibly support beating a significant number of the issues tormenting Blockchain improvement. By adjusting and evolving ideas, methods, instruments, and thoughts that have recently been created in computer programming to this new programming innovation, programmers have a phenomenal chance to start investigating a point that is both basic and fresh out of the plastic new. This paper moves forward toward this path by introducing a complete model of associations between customary programming and the Blockchain climate, including Class graphs, Statecharts, US outlines,

Arrangement charts, Brilliant Agreements charts, and an overall plan for overseeing BOS improvement processes, just as a viable illustration of a paradigmatic Blockchain Shrewd Agreement executing a democratic framework. Our discoveries may be enormously useful to Blockchain ventures, eminently ICOs, who wish to acquire an upper hand by carrying out SE (BOSE) philosophies from the beginning.

References

[1] G. Sachs, "Blockchain-The new technology of trust," *Retrieved April*, vol. 11, p. 2018, 2018.
[2] F. Asuncion *et al.*, "Connecting Supplier and DoD Blockchains for Transparent Part Tracking," *Blockchain: Research and Applications*, p. 100017, 2021, doi: https://doi.org/10.1016/j.bcra.2021.100017.
[3] V. Shermin, "Disrupting governance with blockchains and smart contracts," *Strategic Change*, vol. 26, no. 5, pp. 499–509, 2017.
[4] V. Gatteschi, F. Lamberti, C. Demartini, C. Pranteda, and V. Santamaria, "To blockchain or not to blockchain: That is the question," *IT Professional*, vol. 20, no. 2, pp. 62–74, 2018.
[5] P. Tasatanattakool and C. Techapanupreeda, "Blockchain: Challenges and applications," in *2018 International Conference on Information Networking (ICOIN)*, 2018, pp. 473–475.
[6] A. Alammary, S. Alhazmi, M. Almasri, and S. Gillani, "Blockchain-based applications in education: A systematic review," *Applied Sciences*, vol. 9, no. 12, p. 2400, 2019.
[7] M. Turkanović, M. Hölbl, K. Košič, M. Heričko, and A. Kamišalić, "EduCTX: A blockchain-based higher education credit platform," *IEEE Access*, vol. 6, pp. 5112–5127, 2018, doi: 10.1109/ACCESS.2018.2789929.
[8] F. Agustin, S. Syafnidawati, N. P. Lestari Santoso, and O. G. Amrikhasanah, "Blockchain-based Decentralized Distribution Management in E-Journals," *Aptisi Transactions On Management*, vol. 4, no. 2, pp. 107–113, 2020.
[9] H. Nusantoro, P. A. Sunarya, N. P. L. Santoso, and S. Maulana, "Generation Smart Education Learning Process of Blockchain-Based in Universities," *Blockchain Frontier Technology*, vol. 1, no. 01, pp. 21–34, 2021.
[10] F. P. Oganda, U. Rahardja, Q. Aini, M. Hardini, and A. S. Bist, "BLOCKCHAIN: VISUALIZATION OF THE BITCOIN FORMULA," *PalArch's Journal of Archaeology of Egypt/Egyptology*, vol. 17, no. 6, pp. 308–321, 2020.
[11] T. Salman, R. Jain, and L. Gupta, "A Reputation Management Framework for Knowledge-Based and Probabilistic Blockchains," in *2019 IEEE International Conference on Blockchain (Blockchain)*, 2019, pp. 520–527. doi: 10.1109/Blockchain.2019.00078.
[12] U. Rahardja, Q. Aini, and S. Maulana, "Blockchain innovation: Current and future viewpoints for the travel industry," *IAIC Transactions on Sustainable Digital Innovation (ITSDI)*, vol. 3, no. 1, pp. 8–17, 2021.
[13] J. Yli-Huumo, D. Ko, S. Choi, S. Park, and K. Smolander, "Where is current research on blockchain technology?—a systematic review," *PloS one*, vol. 11, no. 10, p. e0163477, 2016.
[14] Q. Aini, U. Rahardja, and A. Khoirunisa, "Blockchain Technology into Gamification on Education," *IJCCS (Indonesian Journal of Computing and Cybernetics Systems)*, vol. 14, no. 2, pp. 1–10, 2020, doi: 10.22146/ijccs.53221.
[15] Q. Aini, N. Lutfiani, F. Hanafi, and U. Rahardja, "Application of Blockchain Technology for iLearning Student Assessment," *IJCCS (Indonesian Journal of Computing and Cybernetics Systems)*, vol. 14, no. 2, 2020, doi: 10.22146/ijccs.53109.
[16] I. Handayani, R. Supriati, and E. S. N. Aisyah, "Proof of Blockchain Work on The Security of Academic Certificates," in *2020 8th International Conference on Cyber and IT Service Management (CITSM)*, 2020, pp. 1–5.
[17] H. Nusantoro, R. Supriati, N. Azizah, N. P. L. Santoso, and S. Maulana, "Blockchain Based Authentication for Identity Management," in *2021 9th International Conference on Cyber and IT Service Management (CITSM)*, 2021, pp. 1–8.
[18] R. Bhargava, "Blockchain Technology and Its Application: A Review," *IUP Journal of Information Technology*, vol. 15, no. 1, pp. 7–15, 2019.

[19] S. Yang, Z. Chen, L. Cui, M. Xu, Z. Ming, and K. Xu, "CoDAG: An Efficient and Compacted DAG-Based Blockchain Protocol," in *2019 IEEE International Conference on Blockchain (Blockchain)*, 2019, pp. 314–318. doi: 10.1109/Blockchain.2019.00049.

[20] T. Hardjono, A. Lipton, and A. Pentland, "Toward an Interoperability Architecture for Blockchain Autonomous Systems," *IEEE Transactions on Engineering Management*, vol. 67, no. 4, pp. 1298–1309, 2020, doi: 10.1109/TEM.2019.2920154.

[21] Q. Liu, Q. Guan, X. Yang, H. Zhu, G. Green, and S. Yin, "Education-industry cooperative system based on blockchain," in *2018 1st IEEE international conference on hot information-centric networking (HotICN)*, 2018, pp. 207–211.