Using smart contracts for Agile projects

Margi Hingrajia
B.Tech ICT
DAIICT
Gandhinagar, India
201801014@daiict.ac.in

Mahi Patel
B.Tech ICT
DAIICT
Gandhinagar, India
201801039@daiict.ac.in

JayPrakash Lalchandani
Oncampus Mentor
DAIICT
Gandhinagar, India
jayprakash_lalchandani@daiict.ac.in

***Abstract*—This document presents a smart contract based approach that tries to minimize the challenges that are currently faced by the Agile software development model. In our approach, we try to automatize the development process by using the smart contracts as a tool to verify the developed features via acceptance test cases. After the task approval, the client can also allow the smart contracts to do the required payment to the vendor and introduce penalties and grants based upon the sprint and task completion. In our solution, the role of the Product Owner to verify the correctness of the features and tasks is passed on to the smart contracts which immediately gives the outcome when conditions are satisfied. This further helps to increase the transparency and collaboration amongst the team along with keeping a complete track on all the development records.**

***Index Terms*—Blockchain, Smart Contract, Agile methodology, Solidity, Remix.**

I. Introduction

Software development models are used to manage the software projects accurately and improving the quality of the software. Agile methodology is one such widely used technique for the complex projects since it facilitates constant iteration of development and testing. It allows continuous planning, improvement, team collaboration and flexibility throughout the software development lifecycle. The software development process is divided in the form of tasks and sprints by the Product Owner[PO] and later assigned to the developers of the team. Agile methodology requires the communication to increase with the increase in the complexity of the project. The processes get delayed due to lack of communication, reliance upon a central authority and lack of transparency and security. This in result affects the quality, cost and time of the software project.

In this project, we aim to use the emerging blockchain technology and smart contracts to manage Agile software development projects. The smart contracts can be used to automate the processes like test case verification, verification of acceptance criteria, payment to the developers and charging penalties upon inadequate use of time and cost. We use the sprint model to design the smart contracts. They are used to verify the acceptance tests for each task in the sprint and also to verify the requisite conditions set by the client. Upon the approval of the tasks, based on the performance, the payment is directly done by the client to the software development team. Hence, using smart contracts, transparency, security and immutability is maintained in the whole development cycle.

II. Literature Review

In this project, we aim to manage the Agile Projects with an application of Blockchain Technology and Smart Contracts. This section discusses about the basic understandings of the major concepts involved.

A. Software Project Management

Software Project Management deals with managing the software projects to ensure delivery of a quality product, keeping in mind the time and cost constraints. It is necessary for teams to follow good engineering practices such as following each and every planning and requirement gathering stage to improve software quality, creating a culture of transparency where the decisions are made in an open environment and maintaining a decentrality among the team where each member is considered equal and there are no biases imposed by the project manager.

B. Agile Scrum methodology

Agile methodology allows to manage the project by creating several phases. Here, there is a constant collaboration with stakeholders and improvement is done continuously. The team follows the cycle of planning, executing and evaluating. Scrum agile methodology follows an incremental development model where continuous inspection, rapid process adaptation and transparency plays a major role. The main role in this process is played by the Product Owner[PO] and the developers. The software requirements are structured in the form of user stories which are further discussed by the PO with the developers. These implementation tasks are further structured into sprints(iterative cycles) and the developers then submit the work to PO for further review. In the sprint planning stage, a sprint backlog list is created containing product backlog items that the team should complete during the sprint. The challenge of the user stories elicitation include lack of communication between developers and PO along with breaking down the requirements into user stories. For the evaluation, the PO verifies whether the acceptance test cases are passed and the customer requirements are fulfilled.

In our solution, we aim to reduce the role of PO through automating the evaluation process by using Blockchain

technologies and smart contracts.

C. Blockchain

Blockchain is a distributed database where the information is stored in a digital format. It is shared amongst the nodes of computer network and hence there is no single point of failure. We can easily understand blockchain as a list of data records i.e. blocks that are linked using cryptography. Once the data is stored on the blockchain, it cannot be modified without adding a record. The main advantages that blockchain provides are transparency, security, immutability and decentralization. There is no requirement of a third party for information verification and value exchange.

D. Smart Contracts

Smart contract are just a smarter version of any other contract. They are stored on a blockchain and execute when the predefined conditions are met. They automatically execute an agreement without any third party involvement so that all the participating nodes have an immediate update of the outcomes. These transactions are trackable and irreversible.

These contracts allows trustworthy transactions without any central authority and provide transparency, security and trust, cutting down the cost and time. Hence, we can leverage these advantages to combat the disadvantages of agile. With this, we can also scrape the tradeoff between the communication of the team members and the software quality and complexity. An organization can further scale a software project by resolving the issues such as cultural biases, requirement management, lack of trust and transparency, and management of agreements through blockchain.

E. Solidity and Remix

Smart contracts are implemented using Solidity language. Solidity is a high level programming language similar to C++, Python and Javascript and is designed to target the Ethereum Virtual Machine(EVM). It is used for implementing decentralized applications on Ethereum blockchain.

To set up a Solidity environment, we use Remix IDE. It provides an integrated environment for coding, running and deploying the smart contracts in an user-friendly way. It provides a fast development cycle with its preinstalled plugins, versions along with its intuitive GUI. Refer Appendix for a detailed explanation.

III. Motivating example

Blockchain can be used to remove the barriers of Agile methodology for scaling the projects and increasing its efficiency. Let's say for example in creating a complex application similar to PayTm or GPay which requires a lot of features to be implemented, communication becomes a greater challenge to track the progress of each member in the team. As a team member, one can face issues like when their work will be validated, what member X is doing and which state that task is in, when the other dependent task will be completed so that they can start with their task, how is their work being evaluated, when will the payment be done, etc. As a client too one may wish to track whether all the requirements have been implemented correctly or not, all the criteria and protocols have been checked properly or not, the payment is done adequately according to the work submitted, etc. We try to answer all these questions by using smart contracts for agile methodology.

The current project management tools like Jira, Asana, etc have proved to be a great asset to automate the process of software management. It becomes limited when it comes to doing the payment as when the tasks gets accepted, tracking the status of each work assigned by the client, viewing the status of each task by each team member, automating the acceptance test criteria verification without any intermediaries, requiring less communication for collaboration and tracking the progress of the whole development process.

The core advantages that our solution provides over these management tools are allowing the stakeholders to trace the project progress, automating the payment process whenever the smart contract executes and keeping a record of every single transaction that is accessible to each team members which helps to further reduce the update meetings. It also guarantees that the system is unbiased regarding the task evaluation and judging the team member's performance. The immutability of blockchain ensures that nobody can delete or modify a transaction and hence it create a more transparent and accountable environment in the team. And thus, these factors allows our proposed model to be useful in making Agile process more efficient and reliable.

IV. Model implementation

We aim to model a smart contract system to manage Agile software development projects. We have divided the solution into three major contracts. At the sprint planning stage, the set of tasks and their requisite conditions validated by the stakeholders are defined and written into the master smart contract. Along with it, additional smart contracts are created for specific tasks which include the acceptance test cases and particular deliverables. After the development of the task, the developer provides the task smart contract with the required acceptance criteria values. It is then checked with the ones provided by the testers and if they aren't matched then the developer is asked to improve the code.
If all the conditions are met then the task is set to be accepted and its status is then forwarded to the master smart contract along with the total cost and the time taken to complete the task. The master smart contract then compares the actual price and the time with the expected ones and calculates the penalty for the developer if any of the conditions are not satisfied.
After that, the contract on the client side is executed. When

all the criteria and validations are approved, the Ethers are transferred from the client's account to the software vendor's account.
The flow of the implementation is depicted in Figure: 1. The use case diagram and activity diagram are depicted in Figure: 2.

The main functions in these three contracts are as follows:

1) The smart contract for individual tasks:
   - This smart contract defines all the acceptance criteria and checks whether a particular task is accepted or not.
   - In the sprint, the project team identifies the user stories that needs to be implemented and based on which they assign the tasks to the members. For example, if we consider a user story that includes keeping the record of the vendor's sales on an e-Commerce website, the tasks included in the sprints can be designing the dashboard, developing the charts on the dashboard, developing the database, testing the dashboard, implementing and testing feedback page, etc.
   - The tester sets certain criteria such as line coverage, branch coverage, a key that is an SHA256 code of the json file that includes test cases with their expected outputs, expected price etc. These criteria may vary based upon the project requirement.
   - The tester also sets the expected values to these criteria along with the total cost and time for the task execution.
   - After the complete development of the task, the developer feeds in the values of the criteria specified along with the total cost and the key that is the SHA256 code of the json file that contains the test cases and the output generated while executing the code.
   - Then the expected values and the values fed by the developers are compared by the tester. If the outputs of the specified test cases in the json file are same as the expected outputs, then the keys generated will be the same.
   - If few of the criteria are not matching, then the developer is notified to improve the code accordingly and execute the contract again.
   - If all the criteria match, then the task's status along with its completion time and total cost is sent to the master smart contract.
2) The master smart contract for validation of all the tasks:
   - This smart contract is designed by the product owner to check whether all the tasks are submitted correctly and on time.
   - Once all the tasks defined in the sprint are executed correctly, its status is sent to the master smart contract. Additional protocols and conditions common for all the tasks are being checked.
   - The expected time duration and total cost are defined by the Product Owner for each task.
   - The penalty points for the developers are calculated if the price or the time duration of the task execution exceeds the expected values. These penalty points can be further used to rate the developers' performance.
   - If all the tasks defined in the sprint pass the acceptance phase, then the further process of payment is executed by the client.
3) The smart contract of the client side:
   - This contract is written on the client side to check if the sprint is completed correctly on time.
   - The client enters the expected date of completion of the set of tasks defined in the sprint.
   - Based on the sprint completion time received from the master smart contract, the delay is calculated from the difference of the expected date and the actual sprint completion date.
   - If there is no delay, the client transfers 1 Ether to the vendor's account else, based on the delayed days, a specific amount is deducted from the payment.
   - This ensures that as and when the tasks are completed successfully, the payment is done immediately.

Figure: 3 displays Remix outputs of three smart contracts discussed. When the task is deployed, task smart contract is executed. Upon meeting all the conditions, the task is automatically accepted and the master smart contract is executed. After verifying the status of each tasks and calculating the penalties of the developers, it proceeds for the payment. To accomplish the payment, client smart contract is executed and based upon the sprint completion time, the Ethers are transferred from client to the vendor's account.
Refer Appendix for a video demonstration of the proposed model.

V. Model Evaluation

Our proposed solution has leveraged the advantages of the blockchain technology to manage an Agile project in a better way. It proves to be more efficient than the existing approaches in the following ways:

- **Reduce scaling challenges due to communication:** This model can mitigate the issues in communication by allowing the team members and the client to be automatically informed of all the activities and the transactions.
- **Automating the processes:** As and when the tasks are completed and accepted, the status is automatically forwarded to the master smart contract. Upon verifying the conditions stated in the master smart contract, it is directly proceeded for the payment in the client's side. So, there is no need for an intermediary person to manage the task workload. This ensures that the tasks are moved to the next stage of execution immediately after the conditions are met, cutting down the total processing time.

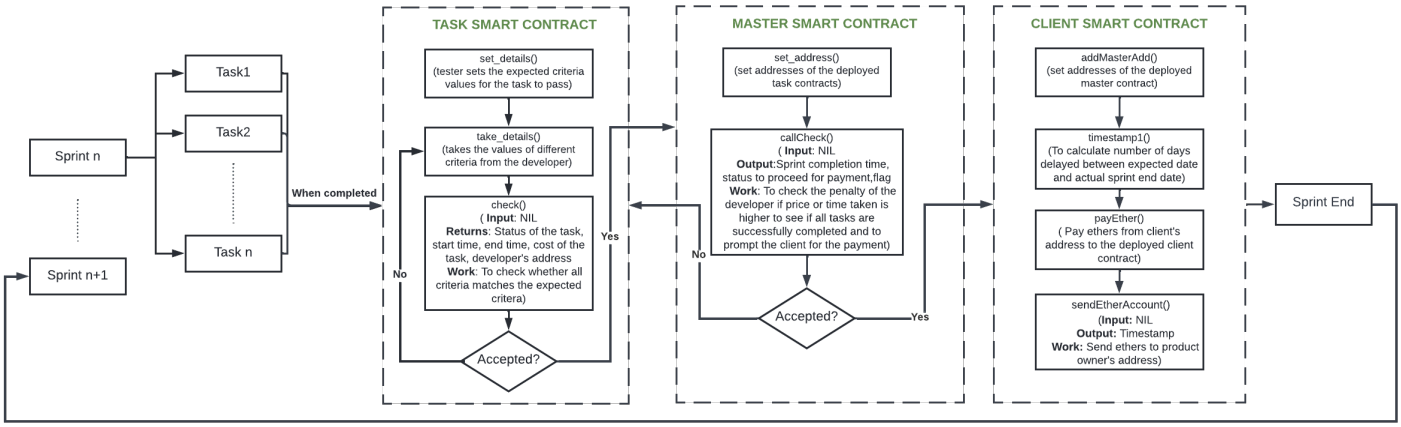

Figure 1. Flowchart of the implemented model.

- **Transparency:** All the transactions on the blockchain get recorded such that any authorized team member can know status of any task or payment at any point of time. This ensures that no information is hidden from anyone and all are on the same page.
- **No single point of failure:** In blockchain, each computer node has the shared copy of the data. So, even if one system fails, it would be very easy to retrieve the data and thus there won't be any data loss.
- **Immutable data records:** Once data gets stored on the blockchain, it is not possible to modify or destroy it. Hence it is easy to retrieve and track the true transaction records without any alterations.
- **Team accountability:** By keeping a complete record of the developments of the sprints, rejections, updates can be easily verified. Each transaction keeps a track of the developers IDs, so they are made accountable for the task they developed. In case of any delayed work, they are directly charged penalties.

VI. Limitations and assumptions

For the implementation of the proposed solution, we have considered the following assumptions:

- The acceptance criteria defined for each of the task directly extracts the value by the code examination and the developer need not enter the same.
- In this dummy template of the proposed solution, we have assumed that the acceptance criteria are measurable in numerical values or can be encoded into a json file.
- The payment done from the client side to the vendor side is in the form of Ethers.
- We have created a dummy model of our proposed solution and hence it contains sample acceptance criteria with its dummy values. The criteria are defined in the form of structure where one can easily delete or add their desired criteria. The values set for the price, time and also the acceptance criteria can be altered as per the need. Adding or deleting the criteria will not affect the processing time, it will just increase the parameters to be set and extracted from the code.

The proposed solution targets to solve majority of the issues but it has some limitations as well.

- The solution requires to create a separate smart contract for each of the task defined in the sprint. This increases the storage overhead. Several copies of information is stored in the blockchain thus increasing the data overhead.
- There are certain criteria such as responsiveness that can't be enumerated into values. So it is difficult to evaluate those and examine the code based on them.
- The model charges penalties based on the disparities of time and cost automatically and hence if there is any delay due to a genuine reason, then the system won't be forgivable, making it inflexible.
- The dummy templates of the contracts needs to be changed according to the tasks assigned in the sprint or when the software overall changes. This might be time consuming for the development team since they have to state different acceptance criteria for each of the tasks.
- The project aims to automate the agile software development process to a greater extent but still some of the phases like the requirement gathering and checking whether all the requirements are converted to the user stories and all the user stories are developed or not requires human intervention.

VII. Conclusion and future work

In our proposed solution we see that smart contracts prove to be optimal choice in order to improve the current agile software management system. We have seen that our model automates multiple processes such as test case verification, payment etc and reduces the intermediary's work. It also strengthens the collaboration and the trust amongst the client and the vendor by increasing the transparency of each transaction made and by making the team more accountable. Additionally, blockchain proves to be an adequate approach

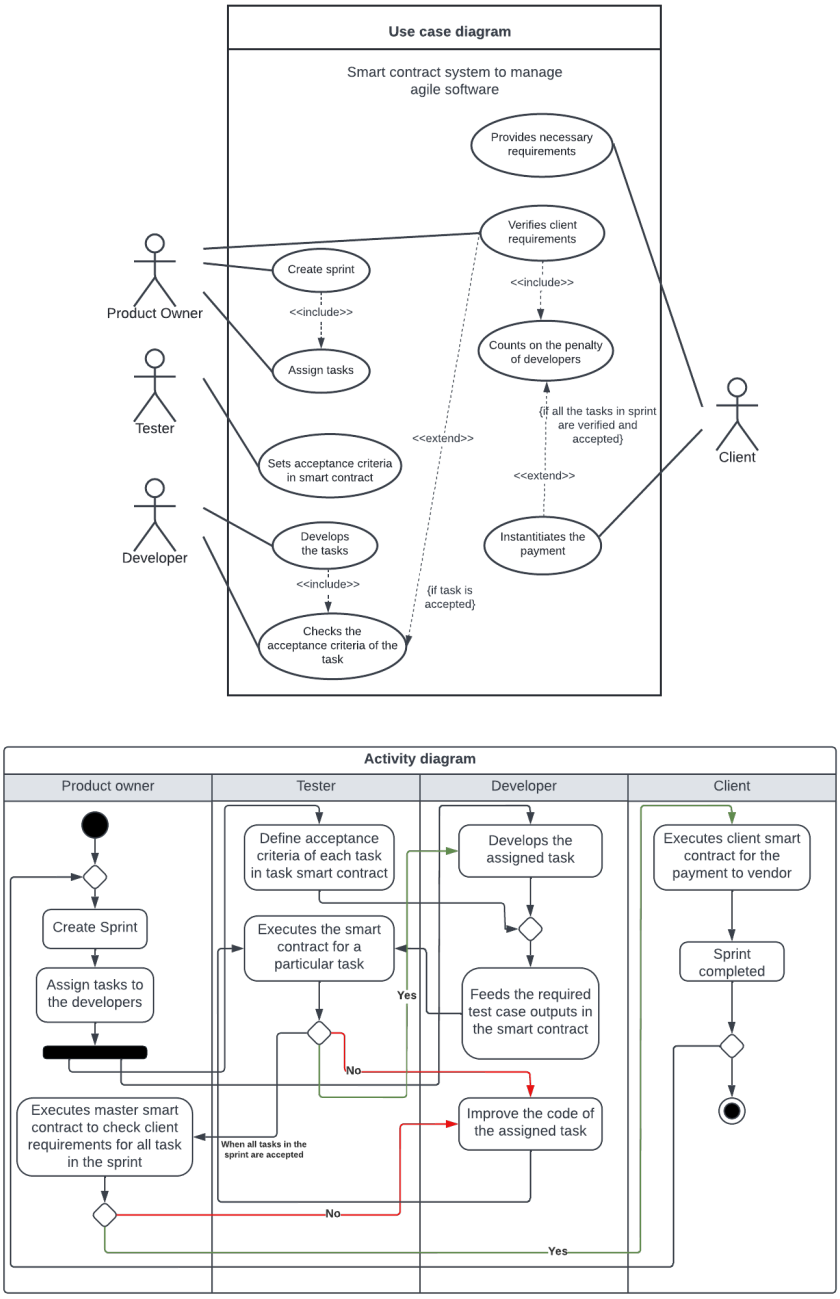

Figure 2. Use case and activity diagram

to track the status of the software and record the workflow as well as the productivity of the team members. Smart contracts also facilitate the payment to the vendor immediately after all the tasks have been approved. Automating the task verification and tracking process, blockchain and smart contracts help to relieve the duties of the Product Owner[PO].

Thus, blockchain helps in maintaining an auditable and complete record of the software project and its transactions. Combining it with the agile software development project will not only increase the collaboration and transparency amongst the teams but also help in scaling the projects to a great extent.

Although the current proposed solution tackles most of the challenges that existed in the agile methodology, but it still does require few improvements and enhancements. The model can be extended to the phases of planning and requirements gathering where we can check upon 100% conversion of the requirements to user stories and also whether all the user stories are developed or not. The acceptance testing can also be improved by adding more

Figure 3. Work flow of the implemented model.

features and criteria for checking the code accurately and thus making it more versatile. We can further adopt certain efficient techniques to improve the storage overhead.

Apart from these, the organizations must understand what all features does blockchain technology brings to the table and how much it is important to make a transition from using conventional agile software project management tools to the blockchain based approach. This project will serve as a basic template which the organization can fit into the repository and can be reused to deal with the client-vendor relationships.

Acknowledgment

We would like to express our sincere gratitude to our on-campus mentor Prof. JayPrakash Lalchandani for his constant guidance and motivation throughout this project. He actively gave us useful insights and feedback on our research. We would extend our appreciation to Prof. Lalchandani for giving us the opportunity to do a research project under his supervision.

References

[1] Valentina Lenarduzzi, Maria Ilaria Lunesu, Michele Marchesi, and Roberto Tonelli. 2018. Blockchain applications for Agile methodologies. In Proceedings of ACM Conference (XP'18). ACM, New York, NY, USA, 3 pages.

[2] Jinal S.Patel. 2020. Generating trusted coordination of collaborative software development using Blockchain. Arizona State University, AZ, USA, 77 pages.

[3] Demi, S.; Colomo-Palacios, R.; Sánchez-Gordón, M. Software Engineering Applications Enabled by Blockchain Technology: A Systematic Mapping Study. Appl. Sci. 2021, 11, 2960.

[4] Demi, S. Blockchain-oriented requirements engineering: A framework. In Proceedings of the 2020 IEEE 28th International Requirements Engineering Conference (RE), Zurich, Switzerland, 27 August–4 September 2020; pp. 428–433.

[5] S. Porru, A. Pinna, M. Marchesi and R. Tonelli, "Blockchain-Oriented Software Engineering: Challenges and New Directions," 2017 IEEE/ACM 39th International Conference on Software Engineering Companion (ICSE-C), 2017, pp. 169-171

[6] R. Liu, H. Subramanian. 2019. Smart Contracts Based Agile Software Development, IEEE Blockchain Technical Briefs: https://blockchain.ieee.org/technicalbriefs/june-2019/smart-contracts-based-agile-software-development

[7] Alex Myers. The Complementary Relationship Between Agile and Blockchains: https://www.nexient.com/insights/the-complementary-relationship-between-agile-blockchains

[8] Solidity documentation: https://docs.soliditylang.org/en/v0.8.13/

[9] 2021. How Is Blockchain Used in the Software Development Industry? : https://www.asynclabs.co/blog/blockchain-development/how-is-blockchain-used-in-the-software-development-industry/
[10] Tariq, F.; Colomo-Palacios, R. Use of blockchain smart contracts in software engineering: A systematic mapping. In Computational Science and Its Applications–ICCSA; Misra, S., Gervasi, O., Murgante, B., Stankova, E., Korkhov, V., Torre, C., Rocha, A.M.A.C., Taniar, D., Apduhan, B.O., Tarantino, E., Eds.; Springer International Publishing: Cham, Switzerland, 2019; pp. 327–337.
[11] Yilmaz, Murat Tasel, Serdar Tüzün, Eray Gulec, Ulas O'Connor, Rory Clarke, Paul. (2019). Applying Blockchain to Improve the Integrity of the Software Development Process. 10.1007/978-3-030-28005-5_20.
[12] P.Rek, Muhamed Turkanvic. 2021. Data modelling for Blockchain Oriented Software Engineering, 2021 Proceedings of the Central European Conference on Information and Intelligent System, Croatia.
[13] Solidity tutorial: https://www.youtube.com/playlist?list=PLL5pYVd8AWtT6vcvLo6vHv3EmtsqiQBxy
[14] Solidity tutorial: https://www.youtube.com/playlist?list=PLbbtODcOYIoE0D6fschNU4rqtGFRpk3ea
[15] Remix IDE documentation: https://readthedocs.org/projects/remix-ide/downloads/pdf/latest/
[16] Agile software development automated by blockchain smart contract: https://youtu.be/WgB9kBALWHI

VIII. Appendix

1) Solidity and Remix guide: https://drive.google.com/file/d/1-P2cJBmnPtCFjSqcJMRIlYEdfnImhhqb/view?usp=sharing
2) Introduction to Solidity and Remix IDE: https://youtu.be/T39iIZfRep4
3) Demonstration of the proposed model: https://youtu.be/s_8xFcMqvDM