



Using Smart contracts for Agile Projects

Presented by: Margi Hingrajia (201801014)
Mahi Patel (201801039)

On-campus mentor: Prof. Jayprakash Lalchandani
(jayprakash_lalchandani@daict.ac.in)

Introduction

- **Agile methodology** is a widely used technique for the complex projects since it facilitates constant iteration of development and testing.
- It requires **increased communication** with the **increase in the complexity** of the project.
- Due to lack of transparency, decentrality and communication, the quality of software produced gets affected.
- **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.
- It helps to maintain **transparency, security and immutability**.

Motivating examples

Security Breach:

Violation of the protection of personal data in Jira or Confluence – How to deal with data leaks & breaches in a GDPR-compliant way

By Ruben Jäckle - 3. March 2022 - Reading time: 5 minutes

NEWS THREATS

Repository Blunder! GitHub Data Leak Incidents Impact Over 200,000 U.S. Patients

By CISOMAG - August 18, 2020

src: <https://cisomag.eccouncil.org/9-leaky-github-repositories/>
<https://actonic.de/en/violation-of-the-protection-of-personal-data-in-jira-or-confluence/>

Lack of transparency:

The [National Audit Office \(NAO\)](#)'s review of the Universal Credit project clearly indicated that this was in fact mainly a failure of Agile adoption, with procurement and contracting secondary reasons. A **lack of transparency** and inadequate performance problems. All these, of course, can be summed up as Agile failures.

src: <https://content.intland.com/blog/agile/failure-of-agile>

Lack of communication and coordination:

Project X; A scheduling system for a large organisation in the energy sector

Communications between teams, and particularly between vendors was fractured and non-time sensitive. Standups were not compliant to Agile practices and on top of that,

src: <https://www.4mation.com.au/blog/3-failed-agile-projects-and-where-they-went-wrong/>

Delays:

Delays getting client to sign-off on acceptance tests – this is true not only for sign-off but also getting client time for resolving requirement issues, give feedback on demos etc.

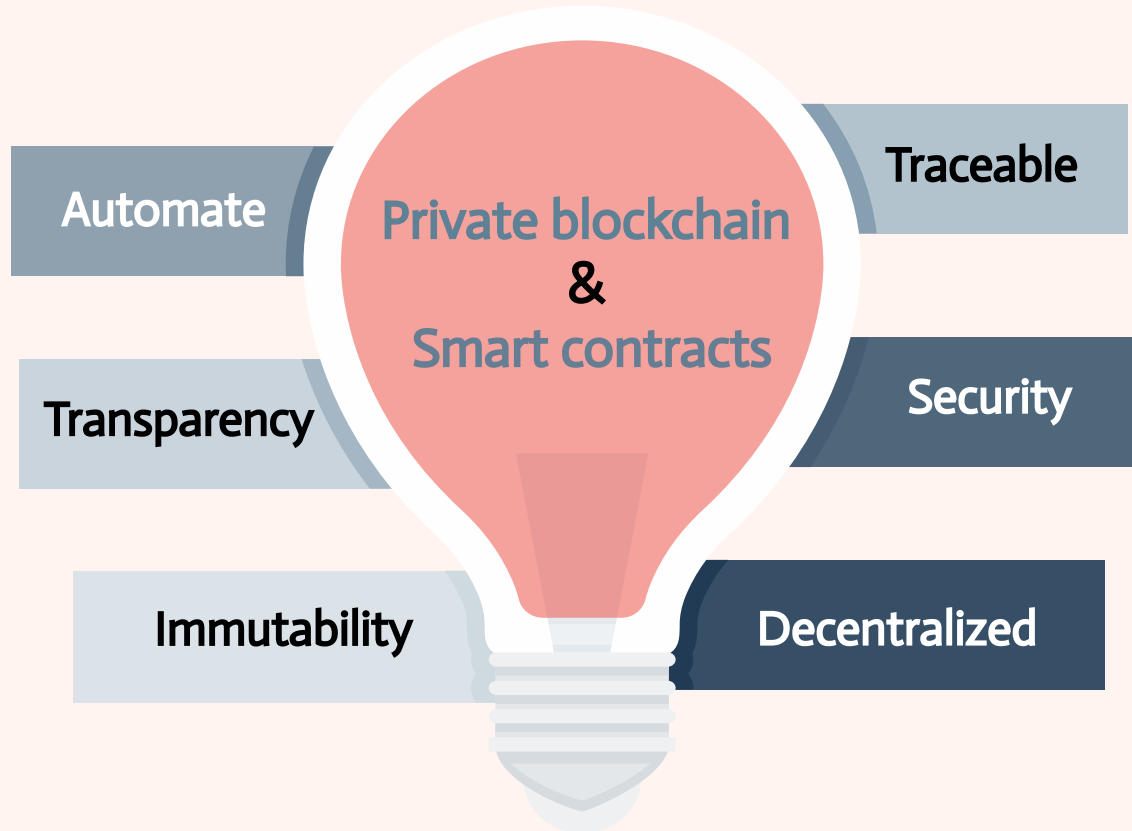
Project approvals - waiting for projects to get approved with developers sitting around thus leading to waste of time and money.

src: <https://www.infoq.com/news/2009/12/agile-project-delays/>



FAIL

Motivation to use smart contracts



Objective

- To check the use of smart contract for Agile projects
- To automate the acceptance testing process by smart contracts.
- To achieve transparency, traceability and security by using **private blockchain**.
- To relieve the duties of **Project Owner(PO)**.

Literature Review

Agile methodology

It follows an **incremental development model** where continuous inspection, rapid process adaptation and transparency plays a major role.

Smart 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.



Software Project management

It deals with **managing the software projects** to ensure delivery of a quality product, keeping in mind the time and cost constraints.

Blockchain

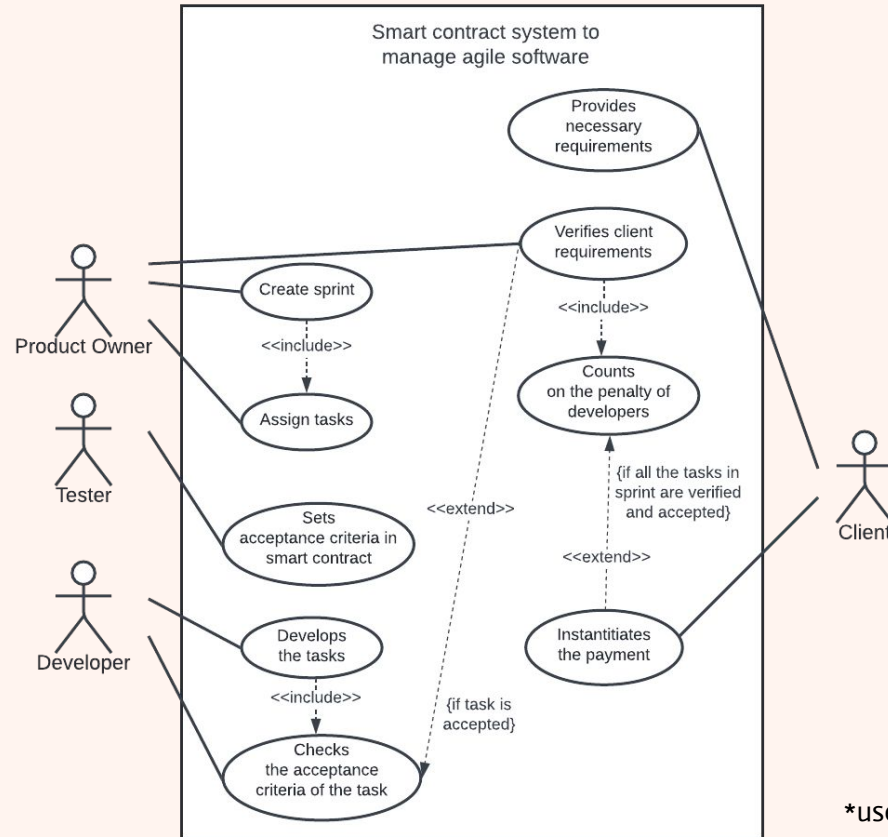
It is a **distributed database** where the information is stored in a digital format. It is shared amongst the nodes of computer network.

Solidity and Remix

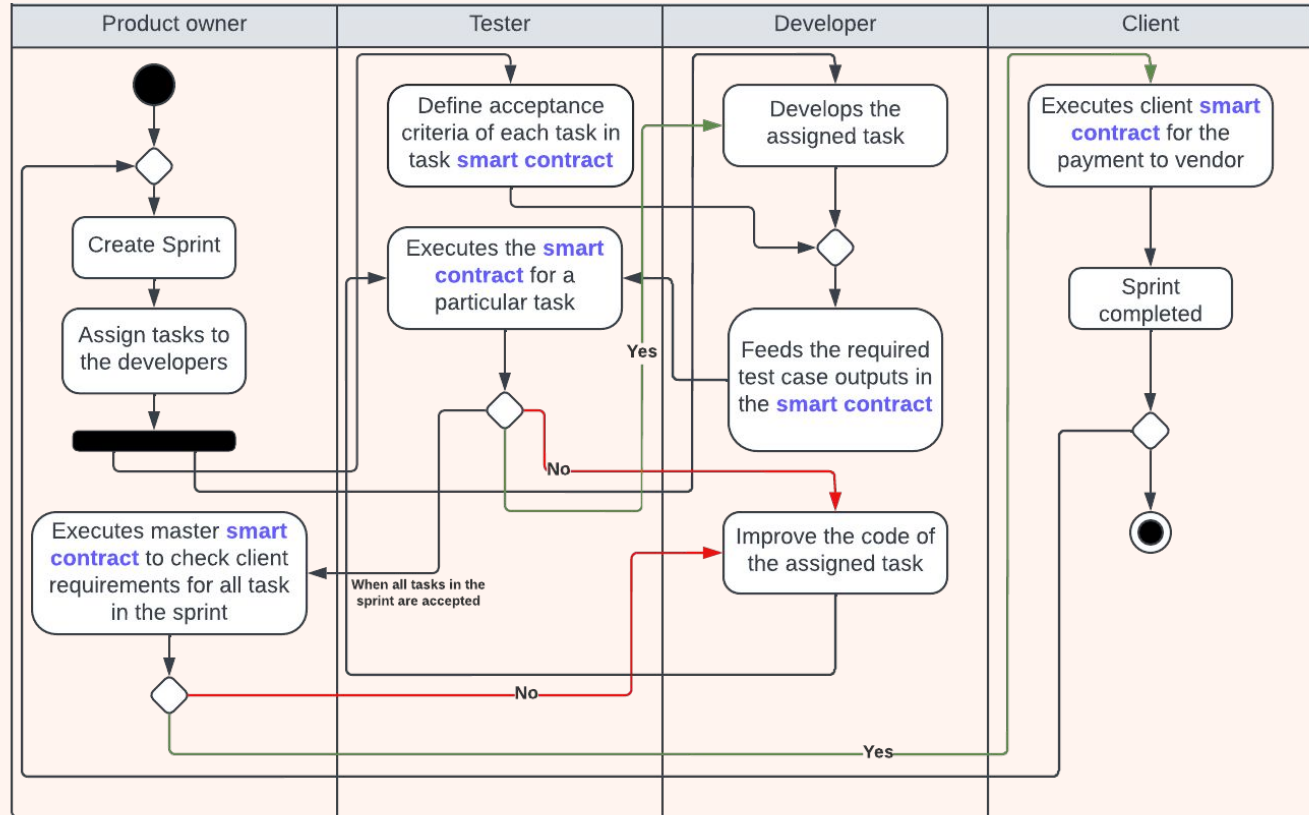
Solidity is a high level programming language used for implementing **decentralized applications** on Ethereum blockchain.

Remix provides an integrated environment for coding, running and deploying the **smart contracts**.

Key use-cases*



Role based activities

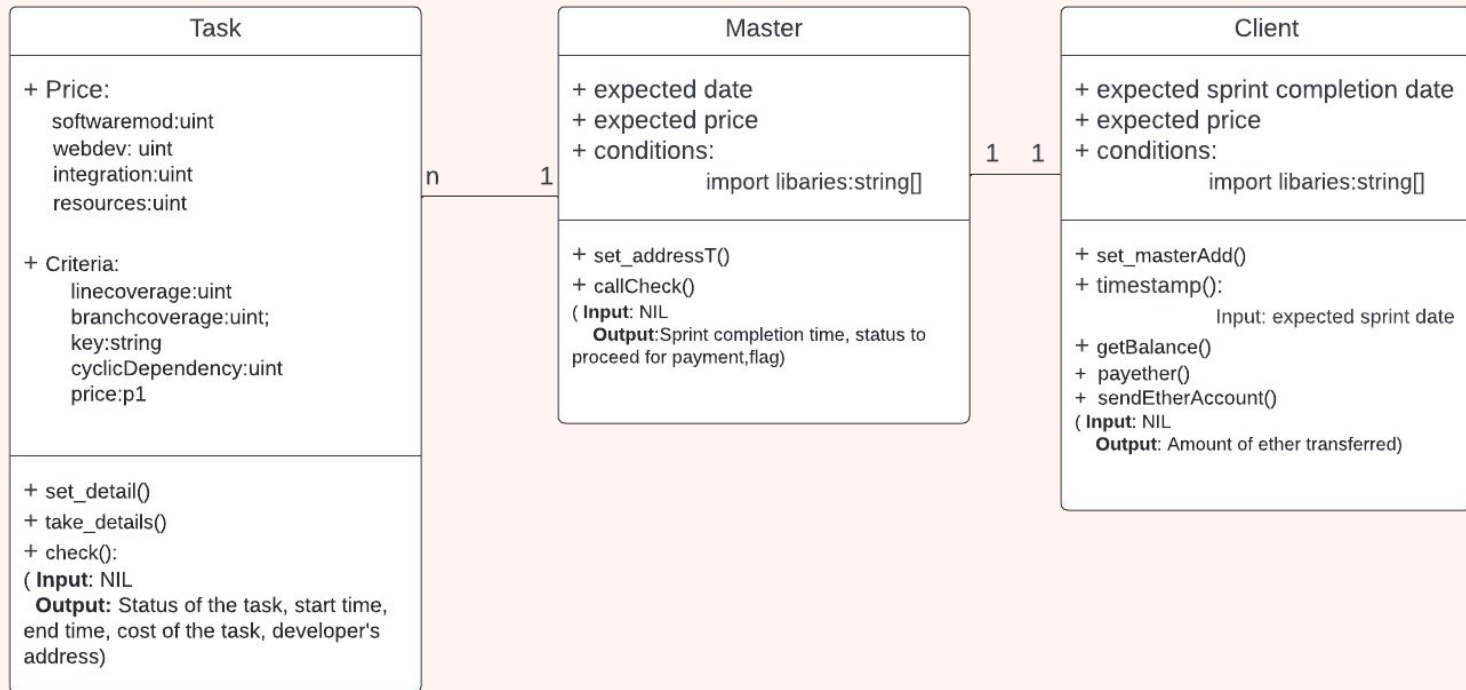


Prototype Implementation

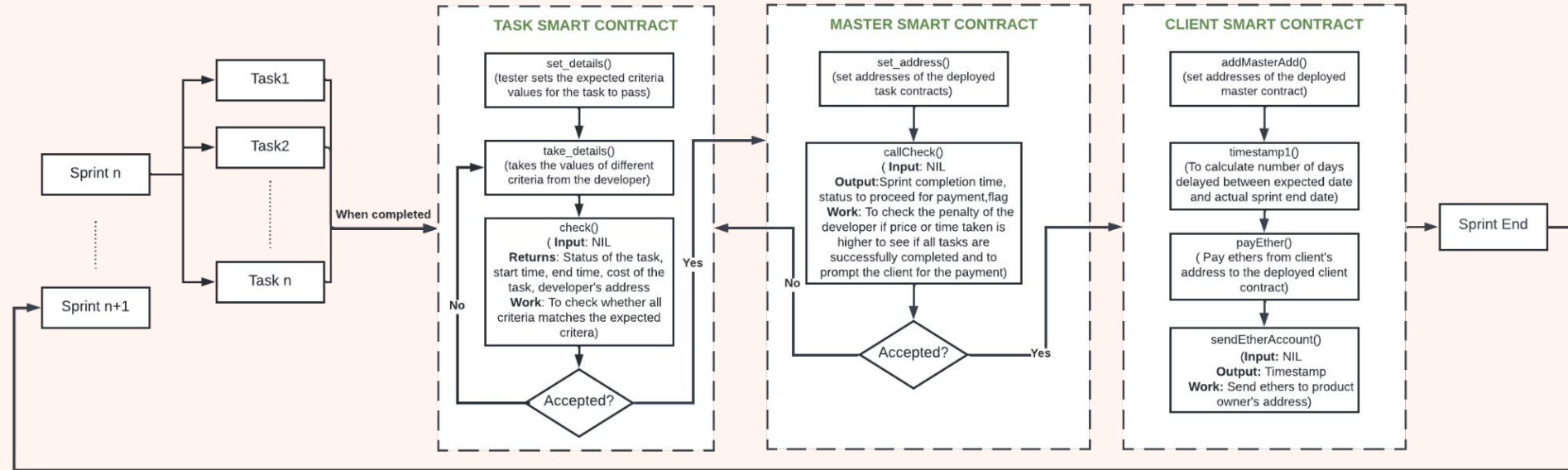
We have divided the solution into three major contracts :

1. **Task smart contract**: Checks whether a particular task of a sprint satisfies the acceptance criteria given by the tester or not.
2. **Master smart contract**: Checks the terms and conditions for each task given by the client and calculates penalty for the developers if any task is overdue.
3. **Client smart contract**: As and when the set of tasks are approved, it executes the payment based on the given deadline.

Key classes



Workflow in agile using Smart Contracts



Screenshots of working prototype

TASK SMART CONTRACT

DEPLOY & RUN TRANSACTIONS

At Address Load contract from Address

Transactions recorded 3

Deployed Contracts

▼ TASK AT 0XA13...EAD95 (MEMORY)

set_detail

take_detail 60,80,"40ece4500f7575f3"

check

0: string: Accepted
1: uint256: 1650781436
2: uint256: 1650781424
3: tuple(uint256,uint256,uint256,uint256): 60,7
0,80,90
4: address: 0xAb8483F64d9C6d1EcF9b849Ae
677dD3315835cb2

Low level interactions

CALLDATA

Transact



sets the details of the
acceptance criteria



takes the details from the
developer to test



the task is accepted



Screenshots of working prototype

DEPLOY & RUN TRANSACTIONS


> TASK1 AT 0X9EC...DDE84 (MEMORY)  


> TASK2 AT 0X9BF...77DD7 (MEMORY)  

> TASK3 AT 0X99C...C196D (MEMORY)  


▼ MASTER AT 0XD91...39138 (MEMORY)  

callCheck

setAddressT 

penalty 

0: uint256: 3

Low level interactions 


CALLDATA
Transact

set task contract
addresses


to view the penalty
of developers


MASTER SMART CONTRACT


listen on all transactions ☐ Search with transaction hash or address


✓ [vm] from: 0x583...eddC4 to: master.callCheck() 0xd91...39138 value: 0 wei data: 0x4a6...abcda logs: 0
hash: 0x4ce...d9973 **Debug** 


status true Transaction mined and execution succeed


transaction hash 0x4cec2c2ebc2dd3d695ea36426403c164dccb82781778ad10ef4571011acd9973 


from 0x5838Da6a701c568545dCfcB03FcB875f56beddC4 


to master.callCheck() 0xd9145CCCE52D386f254917e481e844e9943F39138 

gas 1247255 gas 

transaction cost 1084569 gas 


execution cost 1084569 gas 

input 0x4a6...abcda 

decoded input {} 

decoded output

```
{  
  "0": "uint256: 1650782558",  
  "1": "string: Proceed for payment",  
  "2": "int256: 1"  
}
```



Screenshots of working prototype

CLIENT SMART CONTRACT

The screenshots show the 'DEPLOY & RUN TRANSACTIONS' interface for a 'JavaScript VM (London)' environment. The interface includes fields for ACCOUNT, GAS LIMIT, VALUE, and CONTRACT, along with buttons for 'Deploy', 'payEther', 'sendEtherAcc...', 'timestamp1', 'getBalance', and 'Transact'.

Annotations and labels:

- 2 ethers are transferred from client's address to client contract (points to VALUE field)
- add master contract's address (points to addMasterAdd button)
- to pay ether to the contract (points to payEther button)
- to transfer ether to vendor's address (points to sendEtherAcc... button)
- client enters expected date of task completion (points to timestamp1 field)
- get the balance of contract's account (points to getBalance button)
- ethers successfully transferred (points to the account balance display in the third screenshot)
- expected date (points to timestamp1 field in the fourth screenshot)
- account balance updated (points to the account balance display in the fourth screenshot)

Assumptions for the prototype implementation

- In the [smart contract template](#), we have assumed that the acceptance criteria are measurable in numerical values or can be encoded into a json file.
- For every task, there will be a [client consensus](#) to decide upon the feature prototype and the requisites.
- The payment done from the client side to the vendor side is in the form of [Ethers](#).
- The smart contract template contains sample acceptance criteria. The criteria are defined in the form of structure where one can easily delete or add their desired criteria.

Limitations

- 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 redundancy**.
- With the linear increase in the storage data, it would become difficult to scale.
- It is difficult to automate the complete agile software development process.

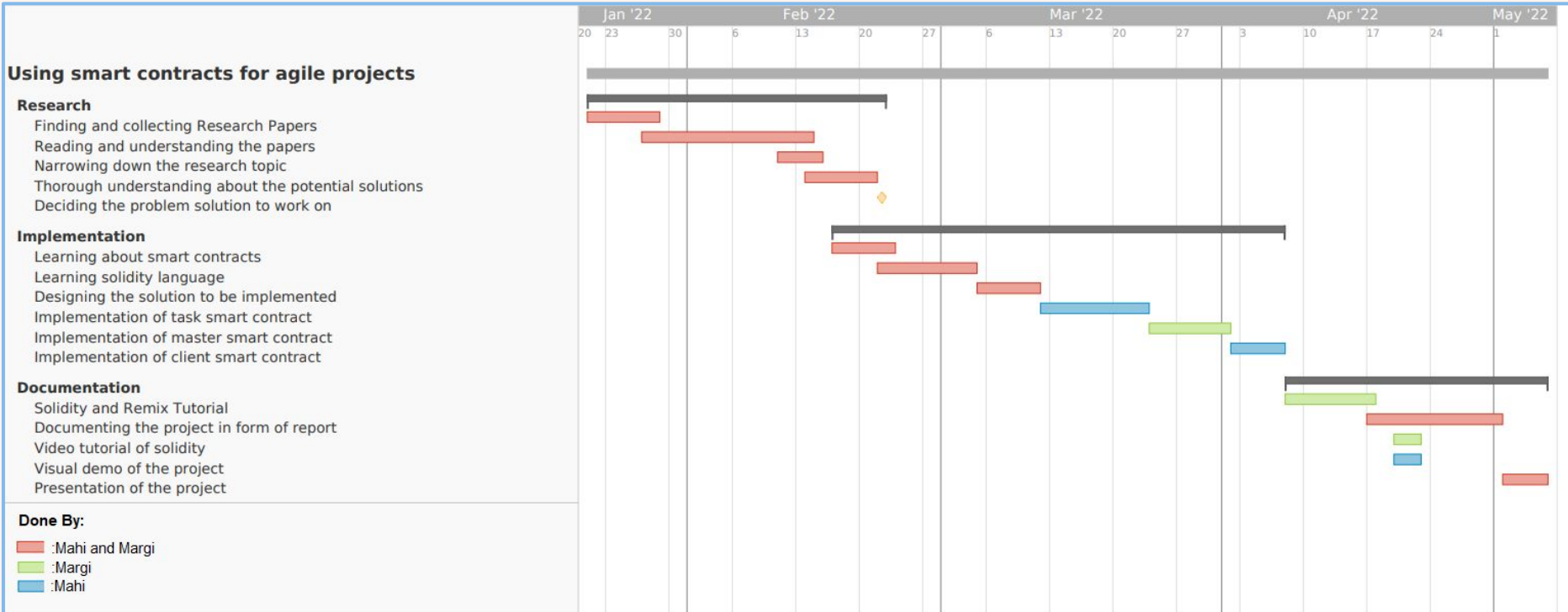
Future Work

- The model can be extended to the phases of planning and requirements gathering. We can implement task prioritization and automate the task distribution.
- We can create a web portal and a DApp which acts as a frontend to communicate with users.
- To reduce the data storage overhead, we can use the off-chain storage facilities.
- This prototype implementation serves as a support to an idea for which we see a brighter scale.

Conclusion

- Smart contracts prove to be optimal choice in order to improve the current agile software management system.
- 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.
- Blockchain helps in maintaining an auditable and complete record of the software project and its transactions.

Project Timeline



Documentation

1. **Solidity and Remix guide:**
<https://drive.google.com/file/d/1-P2cJBmnPtCFjSqcJMRllYEdfnImhhqb/view?usp=sharing>
2. **Introduction to Solidity and Remix IDE:** <https://youtu.be/T39ilZfRep4>
3. **Demonstration of the implemented model:** https://youtu.be/s_8xFcMqvDM
4. **Report:**
https://github.com/mahiz9patel/BTP-Blockchain-in-SE/blob/main/Documentation/201801039_btp.pdf
5. **Github link:**
<https://github.com/mahiz9patel/BTP-Blockchain-in-SE>

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. Md. Shoaib Farooq, Z Kalim, J.N.Qureshi, S.Rasheed, A. Abid, 2022. A Blockchain-Based Framework for Distributed Agile Software Development. IEEE Access, 19 pages.
6. <https://cisomag.eccouncil.org/9-leaky-github-repositories/>
7. <https://actonic.de/en/violation-of-the-protection-of-personal-data-in-jira-or-confluence/>
8. [The Complementary Relationship Between Agile & Blockchains | Nexient](#)
9. [Smart Contracts Based Agile Software Development - IEEE Blockchain Initiative](#)
10. [Solidity, Solidity Tutorial - YouTube](#)
11. [Best Ethereum Solidity beginner level tutorials - YouTube](#)
12. [Slidesgo](#)



Thank You

Benefits of using smart contracts

01

Reduce scaling challenges due to communication

It allows the team members and the client to be automatically informed of all the activities and the transactions.

02

Automating the processes

When the conditions are satisfied, the smart contract gets executed immediately without any intermediary cutting down the total processing time

03

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. It ensures that all are on the same page.

Benefits of using smart contracts

04

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 .

05

Immutable data records

Once data gets stored on the blockchain, it is not possible to modify or destroy it and hence it remains secured from any data breach.

06

Team accountability

By keeping a complete record of the developments of the sprints, rejections, updates can be easily verified.