Camunda Con 2023

Process Modelling of Smart Contracts

Walmart Global Tech India

Table of contents

Introduction

Sushil Anand
Senior Software Engineer
Walmart

Ayush Seth
Senior Manager
Walmart

Where we are

- Adopting a platform-centric method for Business Process Management which could be adopted organization wide.
- Implementing an orchestration layer that connects the user interface with various process transitions.
- Leveraging template-driven strategies for the UI ensures uniformity and a streamlined interface.
- Introducing a management console for template, end-to-end process, and rule orchestration.

Where we are marching towards

- Broaden orchestration across various sectors like smart contracts, ensuring all stakeholders are incorporated.
- The smart contracts demand specific language constructs, develop interoperability between these language constructs.
- Integrate smart contracts/ rules with minimum human touchpoint within the blockchain mechanism

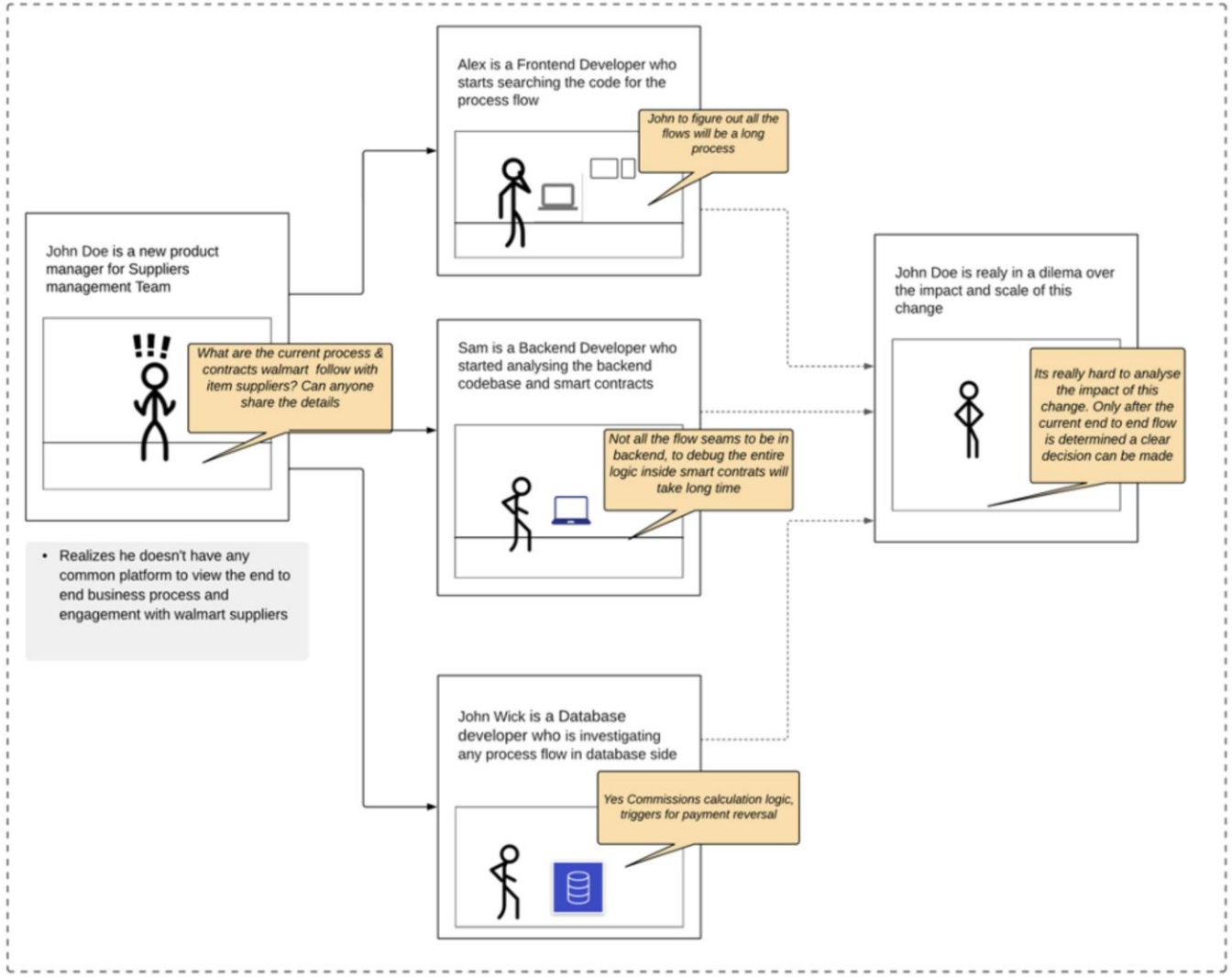
Context

- Organizations are currently using blockchain and smart contracts in many aspects of its business, from supply chain vendor contracts to other contracts in the retail business.
- Blockchain-based systems enable teams to build trustworthy applications, but they also introduce significant new challenges with regards to stakeholder visibility.
- It interestingly brings in a challenge of restricting all the stakeholders to take part in the modelling of the contract.
- Most of the current smart contract programming languages need specialised knowledge of the respective programming constructs.
- If we really focus on just the “contract” part of the smart contract it often deals with plain business rules.
- These business rules are usually driven by the use case which we are trying to solve rather than being specific to any programming constructs.

Problem Statement

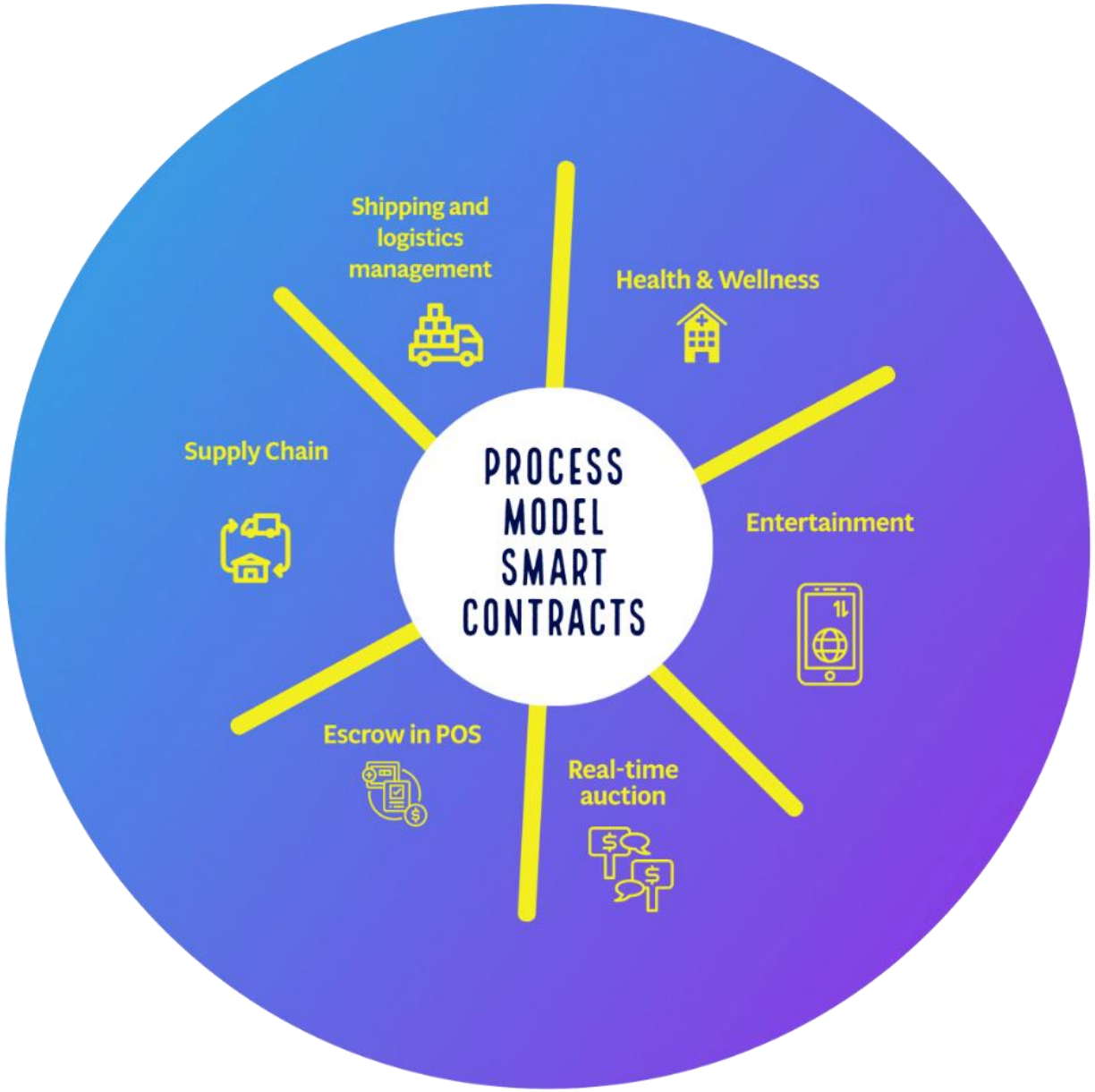
- Every big enterprises are adopting blockchain and smart contracts, yet many are unaware of the impact on its core revenue-generating business processes
- Business facing teams are not always required to know the internal details of the technology aspects used in the solutions.
- In case of end user facing smart contracts, it gets difficult to tie the end-to-end flow in terms of business and how the underlying code and tech is functioning.
- Analyzing impact for any change gets difficult and makes even a small change very critical due to lack of centralized place for the end-to-end flow.
- Smart Contracts are always very business centric so it should be decoupled with the underlying technology and can be flexible in terms of any forward-looking changes we want to make.
- Company’s core business processes nearly always span multiple microservice, making it difficult to gain visibility into the current state of an end-to-end process and to ensure that errors within a process are handled reliably and consistently

Problem Statement



Impact on Revenue-Generating Business Processes

- Blockchain based systems provide teams to build trustworthy applications but also introduce significant new challenges because a company's core business processes nearly always span multiple microservice, making it difficult to gain visibility into the current state of an end-to-end process and to ensure that errors within a process are handled reliably and consistently
- Core and highly revenue generating business processes and rules started to be written in developer centric programming languages like Solidity Rust etc
- Very Less focus on the revenue generating part of the business process
- Huge cost to implement process changes as the process flows are not abstracted



Business Needs

- Ability to visualise and represent the entire flow at a single place.
- Analyse the impact of any small change without getting in the nitty gritty of the code and the programming constructs
- Rapidly bring in new changes at any point of time and easy rolling out across the chain of suppliers.
- Manage visibility while executing a business process.
- Having a standard business contract between the business process, backend microservices, end users and front-end channels.

Solution Overview

1. LOOK AT YOUR BUSINESS PROCESS

Your business process is the revenue generating part of the whole product. Ability to express it as part of the product is the most important thing

2. Model , Collaborate & Transform

Ability to model your contracts , collaborate and transform into deployment compatible code

3. Deploy the same Business process to production

All stakeholders of the product including business, product, engineering speaks the same business process, views the same process and same process goes to production

Process Model your Smart Contract

Domain and Spread

Blockchain

- Blockchains are a new way to efficiently store and access data. Also known as distributed ledger technology, blockchains are publicly available records of transactions that can be accessed by anyone, anywhere.
- A blockchain is thus a shared ledger of transactions where data cannot be altered or deleted once created. It provides a transparent and accessible method to track information, along with the guarantee that the data available is correct and has not been changed.
- Blockchains are typically managed by a peer-to-peer (P2P)computer network for use as a public distributed ledger, where nodes collectively adhere to a consensus algorithm protocol to add and validate new transaction blocks.

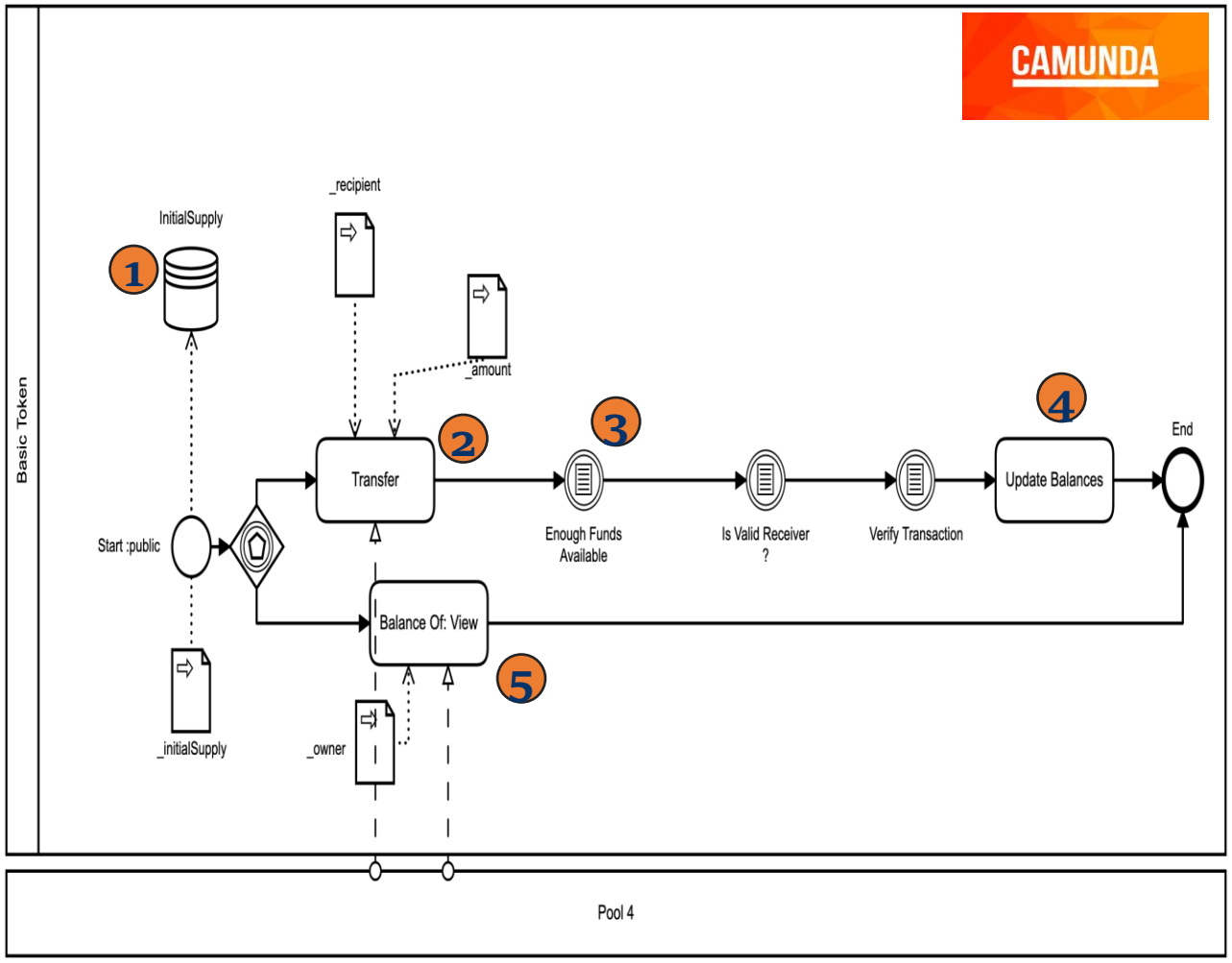
#	Smart Contracts	Process Definitions
1.	Smart contracts are on-chain pieces of programs; these are lines of code deployed on blockchains. Both “smart” and “contract” point to the core characteristics of these on-chain programs.	A BPMN process is a sequence of activities leading from some defined triggering event to one or more possible end states. All possible sequences leading from the start event to some end state are defined by the process model.
2.	They are “smart” because they are fully automated and once its deployed smart contracts do what they are programmed to do without any intermediary.	BPMN once deployed works in a similar automated way to execute in sequence what they are designed to execute.
3.	Furthermore, the “contract” part indicates a binding agreement these pieces of software enforce.	Process Definitions works as an agreement between the client application and the business logic. It ensures the execution always follows the defined path.
4.	Essentially, smart contracts trigger specific predefined actions when certain predefined conditions are met. One of the most popular smart contract programming language is Solidity.	Essentially, Process Definitions are a set of predefined flows which execute sequentially based on specific triggers and input. One of the most popular workflow and decision automation platform is Camunda

Programmatic Smart Contracts

- Consider a smart contract which defines a basic ERC-20 token written in solidity programming language. We call this token “Basic Token”.
- Below are the few things which the contract has
  - InitialSupply - Variable that stores the initial supply of tokens
  - balances - A Mapping that stores the balance of each address that hold the token.
  - transfer - A function that transfers tokens from the senders account to a specified _recipient address. This function needs that the sender has sufficient tokens to transfer, the recipient is not the sender, and the transfer does not cause an overflow.
  - *balanceOf* - Is a function that returns the token balance of a specified _owner address
  - *receive* - A function that is called when the contract receives Ether. It emits the received event.

```
/* SPDX-License-Identifier: GPL-3.0-or-later */
/* Copyright Contributors to the spdx-examples project. */

pragma solidity >=0.7.0 <0.9.0;

contract BasicToken {

    uint public initialSupply;
    mapping(address=>uint) balances;

    constructor (uint _initialSupply){
        initialSupply = _initialSupply;
        balances[msg.sender] = _initialSupply;
    }

    function transfer(address _recipient, uint _amount) public {
        require(balances[msg.sender] >= _amount, "Not enough funds");
        require(_recipient != msg.sender, "No need to send tokens to yourself");
        require(balances[_recipient] + _amount > balances[_recipient]);
        balances[msg.sender] -= _amount;
        balances[_recipient] += _amount;
    }

    function balanceOf(address _owner) public view returns(uint) {
        return balances[_owner];
    }

event Received(address, uint);
receive() external payable {
emit Received(msg.sender, msg.value);
}

fallback() external { x = 1; }
uint x;
}
```

Using Visual Business Processes for Smart Contract




```
/* SPDX-License-Identifier: GPL-3.0-or-later */
/* Copyright Contributors to the spdx-examples project. */

pragma solidity >=0.7.0 <0.9.0;

contract BasicToken {

    uint public initialSupply;
    mapping(address=>uint) balances;

    constructor (uint _initialSupply){
        initialSupply = _initialSupply;
        balances[msg.sender] = _initialSupply;
    }

    function transfer(address _recipient, uint _amount) public {
        require(balances[msg.sender] >= _amount, "Not enough funds");
        require(_recipient != msg.sender, "No need to send tokens to yourself");
        require(balances[_recipient] + _amount > balances[_recipient]);
        balances[msg.sender] -= _amount;
        balances[_recipient] += _amount;
    }

    function balanceOf(address _owner) public view returns(uint) {
        return balances[_owner];
    }

event Received(address, uint);
receive() external payable {
emit Received(msg.sender, msg.value);
}

fallback() external { x = 1; }
uint x;
}
```

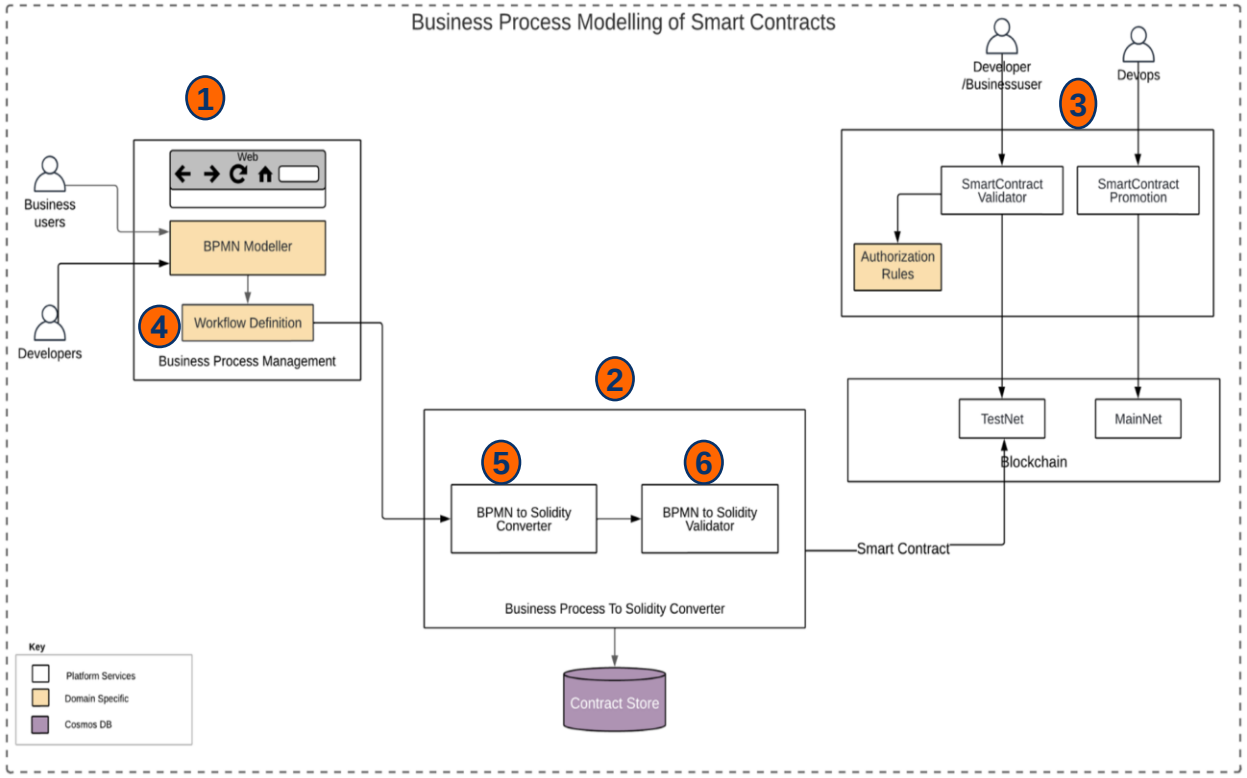
System Design

When converting Business Process Model and Notation (BPMN) elements to Solidity, the focus is on translating the workflow and logic of a BPMN diagram into a smart contract. To facilitate this process, you may create custom BPMN elements that represent the unique features and functions of smart contracts. Here are a few custom BPMN elements you might want to create:

- Contract Element: Represents the overall smart contract, including its name, variables, and functions.
- State Variables Element: Represents the storage variables in a smart contract that hold its state and data.
- Function Element: Represents a function in the smart contract, including its name, input parameters, and output.
- Modifier Element: Represents a modifier, which is a reusable piece of code that can be applied to functions to control their behaviour.
- Event Element: Represents an event, which is used to emit notifications to external applications or the blockchain.
-

- Access Control Element: Represents the access control mechanisms, such as roles or permissions, that govern which addresses can interact with specific functions in the smart contract.
- Function Element: Represents a function in the smart contract, including its name, input parameters, and output.
- External Call Element: Represents interactions with other smart contracts or external functions on the blockchain.
- Conditional Gateway Element: Represents a decision point in the smart contract based on conditions, such as comparisons or logical expressions.
- Loop Element: Represents loops or iterations within a smart contract function, which can be used for repetitive tasks or bulk processing.
- Exception Handling Element: Represents error handling and exception management mechanisms in the smart contract, such as revert, require, or assert statements.
-

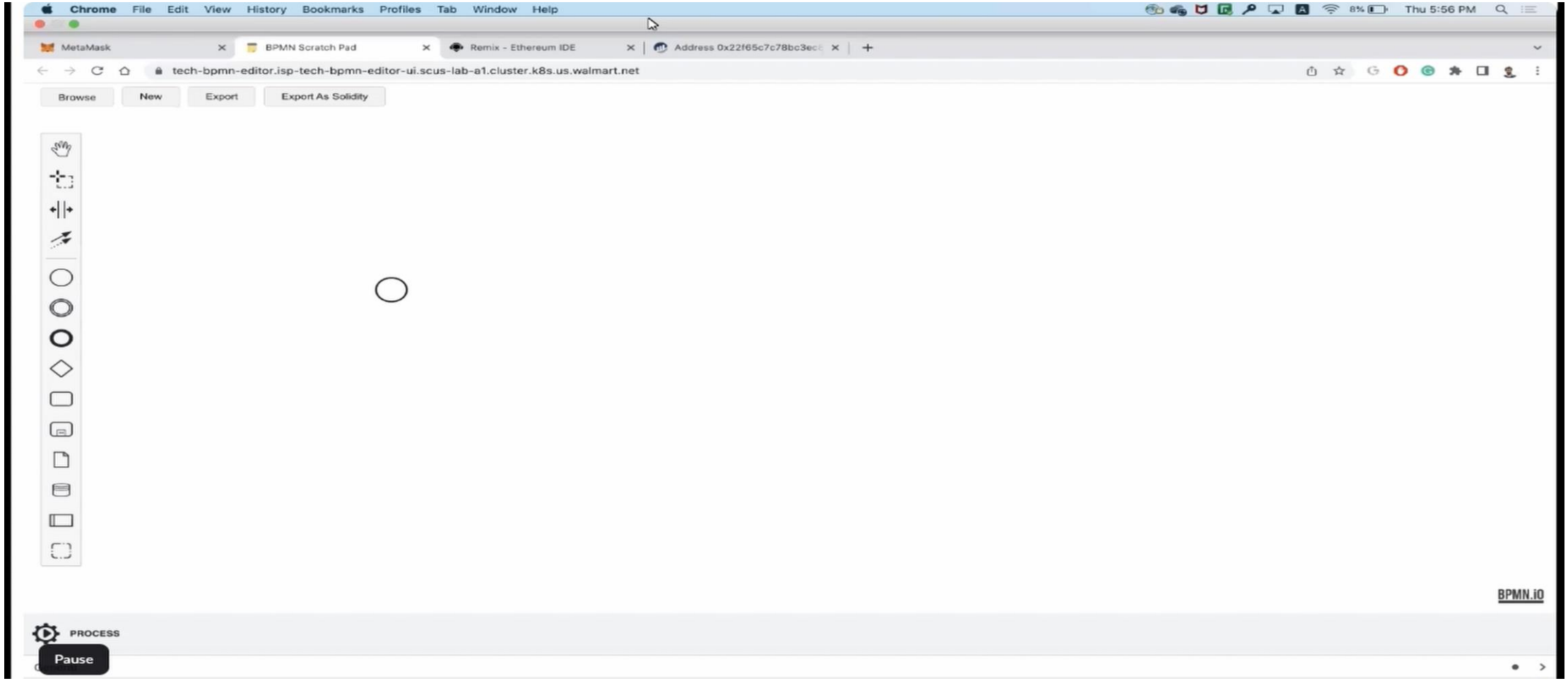
Proposed Architecture



The system will have an **Administrative user interface (Admin Portal)** which will have mainly three panels. The UI will help respective stakeholders to participate in authoring, generation, and deployment.

1. **Process Modelling Panel** – This panel will help the stakeholders in modelling the Business Process. The authored business process could be persisted. The BPMN modelling UI will support custom BPMN elements which are needed for modelling smart contracts.
2. **Contract Generation Panel** – The Contract generation panel will have controls needed to pick any available business process and could be converted to a smart contract. The generated smart contract will be stored in the smart contract store for future use.
3. **Contract Deployment Panel** – The authored smart contracts could be deployed into available blockchain networks. The deployment can be first done in a test-net and then propagated to a main-net.
4. **Workflow Definition** - This layer will parse the business model created by the stakeholders validate and store for future use.
5. **Business Process to Smart Contract converter** – This layer will convert an authored business process into a valid solidity file.
6. **Smart Contract Validator** - This component will validate the generated smart contract and run sanity tests to make sure the auto generated contract. All validated smart contracts will be stored in a smart contract store and the development team can view, edit, and version these files.
7. **APIs for the above**: There will be majorly 3 set of APIs responsible for handling the modelling of business process, smart contract generation and smart contract validation

DEMO


Chrome File Edit View History Bookmarks Profiles Tab Window Help
Thu 5:56 PM
MetaMask
BPMN Scratch Pad
Remix - Ethereum IDE
tech-bpmn-editor.isp-tech-bpmn-editor-ui.scus-lab-a1.cluster.k8s.us.walmart.net
Browse
New
Export
Export As Solidity
BPMN.iO
PROCESS
Pause

Conclusion

Key findings/short comings Possible future direction to build on the work.

- Blockchain based systems provide teams to build trustworthy applications this is one of our key efforts to bring out the company's core business processes which is buried under the realms of coding constructs leading to low visibility across stakeholders.
- Only one critical aspects of this flow is expressed in detail using this paper the finer details of the authoring and custom element needs to be taken up during a detailed implementation.
- Currently we have leveraged Ethereum blockchain for our POCs but the solution could well be extended for any blockchain use cases.
- For smart contracts and choice of programming language we restricted our research to only solidity and this could very well be extended for other smart contract programming languages

Thank you!();

Global Tech India