Blockchain Studio: A Role-Based Business Workflows Management System

Lucie MERCENNE (EIS SEC)
Kei BROUSMICHE (IRT)
Elyès BEN HAMIDA (IRT)
Plan

- **Introduction**
  - Motivation
  - Related Works
- BPMN and Caterpillar
- Proposed architecture
- Case Study
- Conclusion
The Blockchain, a new technology
- Lack of know-how on the market
- Needs to automate business workflows
- Enhance interactions between subject matter experts and technical developers

Motivation: automated Smart Contract generation tool
- Code generation
  - Input: standardized business workflow
  - Output: Smart Contract code
- Code execution
- Complex business workflow
  - In a blockchain, multiple organisations work together
  - Need of Role based access control
Related Works (1/2)

- **Existing solutions**
  - BPMN: a standard Business Process Model and Notation developed by OMG
  - Several works on automated contract generation (e.g., [1] and [2])
    - But no implementation

- **Caterpillar**
  - First solution to cover the whole process (workflow definition, code generation, and execution)
  - Open source project (University of Tartu, Estonia)
  - **Lack of access control and role management**


Plan

- Introduction
- BPMN and Caterpillar
- Proposed architecture
- Case Study
- Conclusion
**Business Process Model and Notation**
- Created in 2004 by the OMG
- Standardized process workflows

**Executable process**
- Task oriented model
- User, script and service tasks
- Stored in a custom XML format (.bpmn)
Limitations

- No access control on Smart Contracts execution
- No roles or organization
Plan

- Introduction
- BPMN and Caterpillar
- Proposed architecture
- Case Study
- Conclusion
Our contribution

Proposed Enhancements

User Interface
- Modeler
- Dashboard
- Viewer

Back-end (off chain)
- Parser
- Controller

Smart Contracts (on chain)
- Generated Smart Contracts
- Role management Contract

Data Base
Our contribution

Proposed Enhancements

User Interface
- Modeler
- Dashboard
- Viewer

Back-end (off chain)
- Parser
- Controller

Smart Contracts (on chain)
- Generated Smart Contracts
- Role management Contract

Data Base
On-chain role management: architecture

Modification of generated Smart Contracts

**Objective:**
Constrain task execution based on the role of the user

Add on-chain management of roles and organisations

- Development of two new Smart Contracts (AccessManager (AM) and ProxyAccessManager (Proxy))
- Each organisation configures its Smart Contracts to define its roles and access policies
On-chain role management: implementation (1/2)

- **Multiple AccessManager**
  - One AccessManager per organisation
  - Manage the list of users’ roles in their organisations
  - Check the access of the user’s permission regarding a task
  - No interaction with the generated Smart Contracts

- **One single ProxyAccessManager**
  - Plays the role of middleware between the generated Smart Contracts and the AccessManagers
  - Is called by the generated Smart Contracts
  - Handles the list of AccessManagers
  - Transmits the access control to the AccessManager
On-chain role management: implementation (2/2)

- Modification of generated Smart Contracts
  - Call the ProxyAccess Manger
  - In modifiers, to control access

- Modifiers
  - Solidity access control keyword
  - If the conditions defined in the modifier are not met, the transaction is rejected
  - tx.origin: retrieves the sender’s identity

Solidity Modifier Example

```solidity
address owner = tx.origin;
ProxyAccessManager proxy = new ProxyAccessManager(address(proxyAccessManager));
modifier isOnly(bytes32 role, bytes32 org) {
    require(proxy.hasAccess(tx.origin, org, role));
}
function Enter_UserTask_start(uint elementIndex) external isOnly(0x526f6c6531, 0x505341) {
    ...
}
```
Our contribution

- User Interface
  - Modeler
  - Dashboard
  - Viewer
- Back-end (off chain)
  - Parser
  - Controller
- Smart Contracts (on chain)
  - Generated Smart Contracts
  - Role management Contract
- Data Base

Proposed Enhancements
Contribution to the back-end and front-end (1/2)

**BPMN Parser**
1. Additional attribute in BPMN file to account for roles in Smart Contracts

**Controller**
2. Integrate access control mechanism
3. Provide identity information to the viewer
4. Add role selection functionality in the viewer

**Viewer**
5. Allow user authentication before the execution of a task
Contribution to the back-end and front-end (2/2)
Case Study

Use case: car log book traceability over blockchain, with roles management

<table>
<thead>
<tr>
<th>Nom</th>
<th>Rôle</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bob</td>
<td>Garagiste</td>
</tr>
<tr>
<td>Alice</td>
<td>Client</td>
</tr>
<tr>
<td>Carla</td>
<td>Assureur</td>
</tr>
</tbody>
</table>

Car placed on the market

Initialization

Assign OwnerShip

Client

Waiting state

Assureur

Insurance phase

Maintenance phase

Garagiste

Car wreck
Plan

- Introduction
- BPMN and Caterpillar
- Proposed architecture
- Case Study
- Conclusion
Conclusions

- The Blockchain Studio Tools allows automated Solidity contract generation (Caterpillar)
- With access control management
- Based on organisation and their roles

Futur Works

- Integrate a Solidity Development Interface
- Develop a graphical development tool for Solidity
- Dynamic access control
Thank you for your attention