Abstract—The Blockchain technology is currently considered as a major paradigm shift in software architectures and applications. Indeed, it enables the automation of business workflows, using smart contracts, and hence the collaboration between trustless entities, without having to rely on any centralized or third parties. In this context, the automated generation of smart contracts, from high level business process models, has recently received increased interests within the research community. In this paper, we introduce the Blockchain Studio framework, which is a fork of Caterpillar, and whose objective is to enable the generation and automation of role-based business workflows on Ethereum based blockchains. The architecture and implementation of Blockchain Studio are firstly discussed, followed by the analysis of the achieved benefits on a practical use case related to the management of vehicles’ life-cycles and data history. Various research challenges and opportunities are then highlighted.

Index Terms—Blockchain; Smart Contract; Business Process; Workflow; Ethereum; Role-based Access Control.

I. INTRODUCTION

The blockchain technology has recently gained increased interests worldwide in a wide range of application domains, such as in government, finance, insurance, mobility, health and energy. This technology has emerged a few years ago to solve the double spending problem in the Bitcoin cryptocurrency system [1], and is a combination of well-know core technologies, including peer-to-peer protocols, cryptographic primitives, distributed consensus algorithms and cryptoeconomics.

A blockchain can be defined as a trustless, decentralized and continuously growing ledger of data blocks that have been validated by consensus among the participating nodes. Each block contains timestamped data transactions, whose integrity and authenticity are guaranteed at any time thanks to hashing and public-key cryptographic algorithms. Once a new block is verified and written to the ledger, transactions cannot be altered retroactively without the collusion of the network majority. More recently, the emergence of smart contracts has unleashed the potential of blockchain in delivering more values and services for enterprises.

The concept of smart contract was firstly proposed by Nick Szabo in 1994 and is now back under the spotlight with the advent of Blockchain technologies. A smart contract is a digital protocol that verifies, executes and enforces contracts’ terms that have been agreed between parties, without having to rely on third parties. Ethereum [2] was one of the first blockchain technology to have proposed a decentralized and turing-complete language, known as Solidity [3], to enable the development of smart contracts that offer the following properties: observability, verifiability, enforceability and security. Each node participating to the blockchain network has a local Ethereum Virtual Machine (EVM), which can execute smart contracts’ byte codes and maintain their global states.

In this context, smart contracts have recently emerged as a convenient way to digitize, share, automate and enforce the execution of business workflows across a network of untrusted organizations, especially in consortium blockchains [4], without having to rely on any single point of trust or failure entities. A workflow can be defined as an orchestrated and repeatable business process, that comprise a sequence of steps (e.g., tasks, operations, events, interactions, etc.), involving two or more organizations and/or users, and whose objective is to create added values to organizations’ activities.

The integration of inter/intra organizational business workflows can reinforce the collaborations, while maintaining trust, and enable a wide range of new applications, such as securities trading and settlement, insurances, notary services, etc. This has recently motivated growing research efforts to study and develop blockchain-based business workflows management systems [5]–[7]. However, several research challenges still need to be addressed to fully realize the potential of such solutions [8].

In this paper, we introduce Blockchain Studio, a novel role-based business workflows management system. This framework is a fork of the Caterpillar project [7], [9], which was extended to enable the management of organizations’ roles and users accounts, as well as the automated generation of secure smart contracts from high level business workflows specifications. The generated secure smart contracts can be deployed on Ethereum [2] based blockchains, and relies on the concept of modifiers, whose objectives is to restrict the access to only authorized and authenticated users.

The remainder of this article is organized as follows. Section II, surveys the related works and highlights the main motivation of this work. Section III, presents the architecture and features of Caterpillar [7], and discusses its main limitations. Section IV, introduces the architecture and implementation of Blockchain Studio, a fork of Caterpillar [7], enabling role-based business workflows automation on Ethereum based
blockchains. Section V, showcases the benefits and performance of the proposed system using practical case study, and discusses various research opportunities and challenges. Finally, Section VI, concludes the article.

II. RELATED WORKS

The issue related to the generation of smart contracts and the automation of business workflows on blockchains has received increased interests in the past few years.

For example, the use of blockchain and its smart contracts feature is considered in [5] to circumvent the need for centralized trusted parties in collaborative process execution, where process specifications, written using the Business Process Model and Notation (BPMN) standard [10], are translated into executable smart contracts. However, the management of roles and access control, a key requirement for the secure integration of business workflows across organizations, was not addressed.

A similar approach is proposed in [6], where the authors propose a novel method to compile BPMN models into solidity smart contracts. This method relies on an efficient encoding technique to store the current state of the process models using a space-optimized data structure. The experimental evaluation shows the efficiency of the proposed approach in terms of gas consumption and transactions’ throughput. However, the management of roles and access control was not covered.

More recently, the same authors have proposed and demonstrated Caterpillar [7], [9], an open-source business process management systems, including a BPMN to smart contracts compiler. Given a BPMN process specification (written in XML), Caterpillar generates the corresponding solidity smart contracts, which can be instantiated, deployed, executed and monitored on Ethereum based blockchains. However, the access control to the generated smart contracts is not handled, and any user can interact with the on-chain business processes.

This paper aims at addressing the limitations of the above research works by proposing Blockchain Studio, a novel role-based business workflows management system. We believe that this feature is a prerequisite for collaborative process across untrusted entities, where the access to a given business workflow’s steps might be restricted to certain types of users or roles. Instead of reinventing the wheel, our proposed framework relies on Caterpillar [7], [9], which is extended to support the management of roles and access control in the generated smart contracts.

III. CATERPILLAR OVERVIEW

Caterpillar [7] is an on-chain open-source Business Process Management System, based on a closed-source application [6] previously done at the University of Tartu, Estonia. This section will describe the current architecture and features of Caterpillar, and will highlight our main contributions.

A. Architecture

The detailed architecture of Caterpillar is described in Figure 1. Caterpillar includes three main blocks, aside from the storage part.

- **Front-end**: Caterpillar’s User Interface is composed of: a modeling panel to create and submit a BPMN model to the Caterpillar engine; a Configuration Panel to (1) view all the Smart Contracts generated from the previous submitted BPMN models stored in a MongoDB database and (2) deploy an instance of a Smart Contract; an Execution Panel to run the execution of one of the deployed process on the blockchain. The front-end is based on angular and is coded in TypeScript.

- **Back-end and storage**: The off-chain back-end of the application is composed of: a BPMN to Solidity Converter module that collects information from the modeling panel and stores the resulting Process Smart Contracts into the Storage; a controller composed of a Configuration Mediator and an Execution Monitor to communicate with the blockchain and the user interface via a REST API. The back-end is coded in TypeScript.

- **Blockchain (Caterpillar Runtime)**: The on-chain back-end consists in several Smart Contracts (Registry and Process Factories) to manage and monitor the deployment of the Process Smart Contracts. Two types of Process Contract are defined in Caterpillar depending on the nature of BPMN task: Worklist Managers contracts for implementing User Tasks (tasks that need user’s action) and Service Managers for Service Tasks that are based on oracles. At the time of writing, the Service Tasks feature of Caterpillar is not functional.

B. Smart contract generation engine

The core functionality of Caterpillar is its ability to generate Smart Contracts that implement BPMN workflows. This feature is handled by the BPMN to Solidity component as described in the previous section. The user defines a BPMN workflow that is composed of a sequence of steps (e.g., user task, script and service tasks). All the basic features of BPMN, including Tasks Types, gateways and loops are supported. The corresponding BPMN model is stored in XML format. The model is then parsed by the back-end, converted into
Organizational business workflows. A key requirement to enable the secure integration of inter/intra role-based workflows management; which we believe, is a requirement for additional information about the Task, but is used by Caterpillar to store values entered by the user when executing the process on the blockchain. The last two beacons points to the BPMN elements connected sequentially to the Task.

C. Limitations

In its current version, Caterpillar is able to translate complex BPMN models into smart contracts that can be deployed and executed on Ethereum based blockchains. However, any users on the blockchain can interact with the deployed business workflows regardless of their permissions and roles in their respective organizations. Thus, Caterpillar integrates neither access control nor role-based workflows management; which we believe, is a key requirement to enable the secure integration of inter/intra organizational business workflows.

IV. BLOCKCHAIN STUDIO: ARCHITECTURE AND IMPLEMENTATION

Motivated by the aforementioned limitations, we introduce the Blockchain Studio, a fork of Caterpillar, which supports on-chain management of organizations’ users accounts and roles, as well as the automated generation of smart contracts with access control.

A. Proposed Architecture

The modification of Caterpillar to create Blockchain Studio is described in Figure 2.

![Blockchain Studio: On-Chain Smart Contracts Architecture](image)

The modifications stand on four main points:

- include role restrictions to an User Task in a BPMN model;
- include the previous custom BPMN feature to the Caterpillar engine in order to generate Smart Contracts with role restrictions;
- add an on-chain role and organization management (Roles & Access Manager);
- modify the back-end, the front-end and the generated Smart Contract to take into account the role management from the modelization to the execution (BPMN to Solidity, Execution Monitor, Modeling Panel and Execution Panel);

Each of these steps is developed in the following subsections.

1) Role restrictions in a BPMN model: The first step consists in restricting a given task to one or multiple specific roles at the BPMN level. Listing 1 shows the XML output of a BPMN task with our solution that adds a role attribute. The bpmn:userTask beacon corresponds to the description of the general information about the Task (e.g., name, id, priority, etc.). The bpmn:Documentation beacon is originally intended for additional information about the Task, but is used by Caterpillar to store values entered by the user when executing the process on the blockchain. The last two beacons points to the BPMN elements connected sequentially to the Task.

```
<bpnm:userTask id="Task_1" name="User_Task1" roles="PSA.Mechanic,Covea.Insurer">
  <bpnm:documentation>
    <![CDATA[ Task Data ]]>  
  </bpnm:documentation>
</bpnm:userTask>
```

Listing 1. XML output for a BPMN User Task (example)

To include a restriction feature, a new role attribute was created in the XML output which contains the allowed roles, in the form of a list of strings, to access the task and the organization managing the roles.

2) Blockchain based roles management: The role management is based on two types of Smart Contracts created and setup before launching Blockchain Studio. Within the consortium, each actor of the network will match an organization participating in the process. The simplified UML diagram of the Smart Contract architecture can be found on Figure 3.

The on-chain role management includes:

- One AccessManager per organization which handles a role list and a user list. It is able to assign one or more roles per user, and state if a user has the required role with a function hasAccess. None of the AccessManagers communicate directly with the generated Smart Contracts.
- One ProxyAccessManager acts as a liaison between the AccessManagers and the generated Smart Contracts. Its address is known from the latter and is called during an access request; in that case the ProxyAccessManager will redirect the request to the asked organization’s AccessManager.

To setup the role management, the ProxyAccessManager is first deployed, then the instances of the AccessManager are deployed and registered to the ProxyAccessManager. Finally the Users and Roles can be registered to any AccessManager before or during the use of Blockchain Studio.
3) Role-based smart contract generation: Caterpillar uses EJS templates (i.e., Embedded JavaScript Templating) to generate Solidity code from the BPMN parsing. To integrate an access right management within the process execution, it is necessary to limit the execution of the functions corresponding to the User Tasks execution. This limitation is done using modifiers in the User Tasks starting functions.

Modifiers: Modifiers are a Solidity feature which allows to limit a function execution following any condition, this condition being described. The example code in Listing 2 shows the used modifier and how it is implemented in a function.

```solidity
public owner = msg.sender;
public proxy = ProxyAccessManager(proxyAddress);
modifier checkAccess(bytes32 organization, bytes32 role) {
    if (proxy.hasAccess(owner, organization, role) == false) {
        revert();
    }
}

function User_Task_Start([args]) checkAccess(
    organization, role) {
    //Execute Task
}
```

Listing 2. Modifier example and implementation

The modifier `checkAccess(organization, role)` will call the `ProxyAccessManager` `hasAccess(organization, role)` function which grants or denies access to the function. If the access is denied, the function is not executed, and the blockchain transaction handling the call is reverted.

4) Others:

a) Back-end modifications: To generate the Solidity code from the EJS templates, a structure containing all the information parsed from the BPMN file is used. To integrate role management it was necessary to add a role and organization attribute to the structure, which aims to fill the arguments of the modifiers. Another parsing step has also been added to collect the role and organization data from the BPMN files.

b) Front-end modifications: To fill the role and organization data into the BPMN file, the Modeling panel had to be modified. For demonstration purposes, a select box has been added to the Execution Panel to select an address with which the process will be executed on blockchain.

B. Implementation

The development of Blockchain Studio has been done following the order of the steps in the previous subsection. The on-chain Smart Contracts were developed in Solidity using the truffle framework which includes an unit test for the Access Managers’ feature. The off-chain modifications of Caterpillar were done using the same tools as to use Caterpillar (developed in TypeScript and Node.js and using a MongoDB database). The source code might be released on Github in the future.

V. CASE STUDY AND EVALUATION

In order to better shed light on the benefits of Blockchain Studio, we describe in this section a practical case study related to the digitalization of vehicles life-cycle over a consortium blockchain. Finally, we discuss various research challenges and opportunities.

A. Case study: Vehicle Life-Cycle Management

Nowadays, the primary way to keep track of vehicle maintenance and mileage information are paper-based car log books, which might be incomplete, corrupted or missing. As a consequence, used car frauds (e.g., odometer frauds, rolling wreck, etc.) are becoming a major problem worldwide. In this context, various initiatives [11]–[14] have been recently launched to investigate the potential of the blockchain technology in securing vehicles information over a tamper-proof and decentralized ledger.

For instance, we recently investigated the use of a consortium blockchain to digitize, secure and share vehicles life-cycles and data history between various stakeholders, including car manufacturer (e.g., PSA), insurance companies (e.g., Covéa) and car services providers [13], [14]. Ethereum-based smart contracts have been developed from scratch to automate the corresponding business workflows, and to enable collaborations and business interoperability between the aforementioned organizations.

Based on our extensive experience in this domain, which resulted in the development of a practical proof-of-concept [13], we propose in the below sub-sections the redesign and refactoring of our prototype based on the proposed Blockchain...
Studio framework. The achieved benefits are then discussed and the research opportunities highlighted.

B. Case study: Redesign using Blockchain Studio

From the described use case, the organizations’ architecture is described in Table I.

<table>
<thead>
<tr>
<th>Organization</th>
<th>Description</th>
<th>Roles handled</th>
</tr>
</thead>
<tbody>
<tr>
<td>PSA</td>
<td>Car manufacturer</td>
<td>Mechanic$^2$</td>
</tr>
<tr>
<td>Covéa</td>
<td>Insurance company</td>
<td>Customer$^1$, Expert$^3$</td>
</tr>
</tbody>
</table>

Each organization, PSA and Covéa, would handle its own roles and users. The insurance part, including the insurance’s client, and the expert in case of accident would be handled by Covéa, while PSA would handle the various mechanics. For this purpose, we created several fictional users who would be assigned the roles listed before.

From the previous use case and the organization architecture, the BPMN model on Figure 4 is created. The process begins by the car manufacturing. Its data, like the registration number, is initialized, then the owner identity is set. The Waiting State describes the state where the car is used without any problem or noticeable event. If the owner wants to sell the car, a new ownership is assigned. If the car needs repair, the Mechanic updates the mileage and other data related to the intervention in the Maintenance Phase Task, and then the car resumes its life cycle to the Waiting State. Finally, in case of accident, an expert states if the car is in a wreckage state or not. If not, it gets repaired and resumes its life cycle. Otherwise the process ends as the car life cycle.

The numbered User Tasks has been assigned a role matching the numbers in the previous table. Therefore, the Assign Ownership task is restricted to Customer only, Maintenance phase is only for Mechanics, and the Insurance phase allows only Experts.

C. Benefits Assessment

Based on our experience of implementation from scratch of the use case and its redesign using the proposed Blockchain Studio solution, we assess the benefits of the latter approach. It has to be noted that these two approaches do not share the same goals. The full manual implementation is intended to be efficient in terms of gas cost and easily upgradeable by other developers. While the Blockchain Studio is meant to speed up designing and deployment for rapid development.

1) Development time: The manual implementation has been achieved in approximately 2 weeks of development by a developer familiar with Solidity. In comparison, the designing of the same functionalities using Blockchain Studio took about 3 hours for the same developer, including the time to learn BPMN formalism. However, since there is some Solidity code to implement the use case (i.e., defining the attributes of the car log book contract) and JavaScript scripts to define the organizations and roles, we have also experimented with a non-developer (a subject matter expert in the field of car manufacturing). As expected, the time to achieve the task have been extended to approximately 7 hours, with little help from developers.

2) Gas Efficiency: Here we compare the gas cost of the contracts deployment of both scenarios. The car log book contract manually developed uses 5 518 613 gas. The source code generated by Blockchain Studio spends respectively: 872 956 for the proxy manager, 2 325 600 per access manager and 6 694 076 gas units for carbook. The total sums up to 14 543 832 gas, almost the triple of the manually coded contract.

VI. Conclusion

In conclusion, we proposed an extension to the existing Caterpillar project by adding access control ability based on role-based segregation. We first described the architecture and mechanisms of Caterpillar before detailing our contribution. Each part that we have modified and added to the existing system have been presented as well as our methodology. Finally, we have demonstrated the benefits of Blockchain Studio by applying it to a practical use case.

In future works we intend to tackle several topics in order to improve and extend functionalities of the proposed solution:

User experience: in its state, Blockchain Studio requires the user to write code for defining roles and organizations as well as some Solidity code to define contracts’ attributes. This could be prevented by developing further user interfaces letting graphical acquisition of roles, organizations and attributes information. The BPMN parser/controller should also be modified accordingly to take into account these data.
Execution verification: one important challenge for smart contracts is its security level. One way to improve its robustness is to verify that the proposed execution model guarantees some user defined constraints (e.g., “a car cannot be declared as a wreck without the intervention of an insurance expert”). This kind of approach has already been investigated in the field of formal verification applied to smart contracts [15, 16]. The overall idea is to describe (1) the execution model using a formal modeling language (e.g., petri net) along with (2) the failure conditions. Then, a simulator automatically executes all the possible paths and verifies that they all comply with the constraints. Blockchain Studio’s BPMN model can be translated to other formal languages (e.g. [17]). Provided that the constraints are defined by the user, it is then possible to integrate or to interface with a formal verification engine.

ACKNOWLEDGMENT

We are thankful to Orlenys Lopez-Pintado and Marlon Dumas for their help and fruitful discussions on the Caterpillar project. This research work has been carried out under the leadership of the Institute for Technological Research SystemX, and therefore granted with public funds within the scope of the French Program Investissements d’Avenir.

REFERENCES