Caterpillar: A Blockchain-Based Business Process Management System

Orlenys López-Pintado[1] and Luciano García-Bañuelos[1] and Marlon Dumas[1] and Ingo Weber[2]

[1] University of Tartu, Estonia
Orlenys.Lopez.Pintado@tudeng.ut.ee, {luciano.garcia,marlon.dumas}@ut.ee
[2] Data61, CSIRO, Australia
Ingo.Weber@data61.csiro.au

Abstract. This demonstration introduces Caterpillar, an open-source Business Process Management System (BPMS) that runs on top of the Ethereum blockchain. Like any BPMS, Caterpillar supports the creation of instances of a process model (captured in the Business Process Model and Notation – BPMN) and allows users to track the state of process instances and to execute tasks thereof. The specificity of Caterpillar is that the state of each process instance is maintained on the Ethereum blockchain, and the workflow routing is performed by smart contracts generated by a BPMN-to-Solidity compiler. The compiler supports a wide array of BPMN constructs, including user, script and service tasks, parallel and exclusive gateways, subprocesses, multi-instance activities and event handlers. The target audience of this demonstration includes researchers in the area of business process management and blockchain technology.

1 Introduction

Blockchain platforms such as Ethereum[3] allow a set of actors to maintain a ledger of transactions without relying on a central authority and to deploy scripts that can be called by external actors to change the state of the ledger. These features provide basic building blocks for executing collaborative business processes between mutually untrusting parties [4, 1]. However, implementing business processes using the low-level primitives provided by blockchain platforms is cumbersome. In contrast, established Business Process Management Systems (BPMS) provide convenient abstractions for rapid development of process-oriented applications, by taking as starting point a business process model represented for example in the Business Process Model and Notation (BPMN) [3].

This demonstration paper introduces Caterpillar, an open-source BPMS designed from the ground up, with the aim of combining the development convenience of a BPMS with the tamper-proofness of a blockchain platform. Like most contemporary BPMSs, Caterpillar supports the creation of instances of a

[3] https://www.ethereum.org/

BPMN process model and allows managers and process workers to track the state of process instances and to execute tasks thereof. The specificity of Caterpillar is that the execution state of each process instance is maintained on the Ethereum blockchain and the workflow routing is performed by smart contracts generated by a BPMN-to-Solidity compiler [4] covering a large array of BPMN constructs, including subprocesses, multi-instance activities and event handlers.

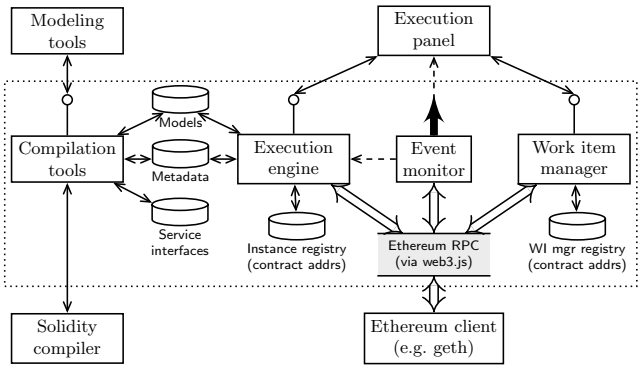

Fig. 1. Architecture of Caterpillar

The architecture of Caterpillar is presented in Fig. 1. The core of Caterpillar consists of the modules inside the dashed rectangle, namely the Compiler Tools, the Execution Engine, the Event Monitor, the Work Item Manager and repositories for process models, metadata, service interfaces and runtime data (specifically, process instance data and work item data).

The Compilation Tools module implements a comprehensive mapping from BPMN to Solidity. Given a BPMN model (in standard XML format), this module generates a smart contract (in Solidity), which encapsulates the workflow routing logic of the process model. Specifically, the smart contract contains variables to encode the state of a process instance, and scripts to update this state whenever a task completes or an event occurs. Caterpillar supports not only basic BPMN control flow elements (i.e. tasks and gateways), but also includes advanced ones, such as subprocesses, multi-instances and event handling. The Compilation Tools module is linked to a Modeling Tool, developed on top of Camunda's BPMN modeler.[4]

Caterpillar's Execution Engine provides operations to deploy a process model (i.e. to deploy the smart contracts generated by the Compilation Tools), to create instances of a deployed process model, to determine which tasks are enabled and to control and record their execution. The execution engine is also responsible for executing automated script tasks and for triggering external service calls whenever a service task is enabled. The Event Monitor listens to events generated by the blockchain and generates notifications whenever a transaction related

[4] https://camunda.org/download/modeler/

to a process instance is validated on the blockchain, so as to keep the other Caterpillar components updated. Caterpillar also provides a default Execution Panel (to create and track process instances) and a default Work Item Manager (to handle user tasks). The functionality of the Caterpillar engine is exposed via a REST API, allowing developers to implement their own execution panel or work list handler instead of using the default ones.

All core modules are implemented in Node.js and rely on standard Ethereum tools to compile the smart contracts (specifically using the Solidity compiler *solc*) and to interact with running instances of the smart contracts via an Ethereum peer node (specifically using the Ethereum client *geth*).

2 Caterpillar Engine and REST API

Caterpillar's Engine provides a REST API that exposes three types of resources: *models*, that is BPMN models and compilation artifacts; *processes*, referring to the set of currently running process instances; and *services*, i.e. references to smart contracts and addresses used for interacting with external services. Table 2 shows the mapping of resource-related actions with the corresponding core module: white background for compilation tools, blue background for execution engine and yellow background for work item manager.

Verb	URI	Description
POST	/models	Registers a BPMN model (Triggers also code generation and compilation)
GET	/models	Retrieves the list of registered BPMN models
GET	/models/:mid	Retrieves a BPMN model and its compilation artifacts
POST	/models/:mid	Creates a new process instance from a given model
GET	/processes/:pid	Retrieves the current state of a process instance
POST	/workitems/:wimid/:wiid	Checks-in a work item (i.e. user task)
POST	/workitems/:wimid/:evname	Forwards message event, delivered only if the event is enabled
POST	/services	Registers an external service
GET	/services	Retrieves the list of registered external services
GET	/services/:sid	Retrieves smart contract/address of an external service

Table 1. Caterpillar's REST API

Once the *caterpillar_core* is running, a user can submit a BPMN model using an HTTP POST request on the URI `/models` with a JSON message that includes the model name and the model serialized in the BPMN 2.0 XML format. To enable the process to interact with external services, a corresponding smart contract/address must have been previously registered using an HTTP POST request on `/services`. This is required because the service's smart contract address is compiled into the process's smart contract.

A new instance of a process can be created using an HTTP POST request on the URI `/models/:mid`, where `:mid` is the identifier provided when the process model was registered. As a result of this request, Caterpillar will return a hyperlink that refers to the newly created process instance in the HTTP header `Location` and an HTTP status 201. Such a hyperlink would have the form

`/processes/:pid`, where `:pid` corresponds to the address assigned by Ethereum to refer to the instance of the process' smart contract.

At any time, the state of a process instance can be queried by using an HTTP GET request on the URI `/processes/:pid`. From here, external actors can obtain the hyperlink to execute a User Task – if enabled. This execution occurs using an HTTP POST request on the URL `/workitems/:wimid/:wiid`. Note that the request is going to a *workitems* component which validates the user access rights before forwarding the call to the process instance contract.

The *execution_panel* illustrates graphically the results of querying the state of a process instance and also allows the execution of user tasks by external actors. Here, every enabled task is visualized in dark green, while activities in execution are drawn in light green. A user may start the execution of an enabled task by clicking on it and submitting the required parameters, as shown in Fig. 2. Only user tasks and catching messages are drawn in the *execution_panel*, because the execution of other types of BPMN elements (e.g. gateways, throwing events, script tasks, etc.) is started internally by Caterpillar. For the demo, we will also use the debug information printed by Caterpillar in the terminal to keep track of internal activity (e.g. execution of script tasks) and interaction with external services.

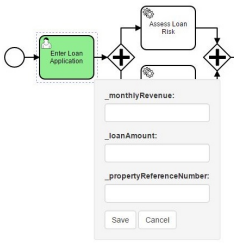

Fig. 2. User task in *execution_panel*

3 Maturity

Caterpillar is in its first release. However, the prototype extends and consolidates ideas presented in previous works [4, 2], with the aim of better aligning the architecture of the BPMS with that of Ethereum. The current version of the prototype supports the following elements in the standard BPMN 2.0: user tasks, service tasks, script tasks, exclusive gateways, parallel gateways, event-based gateways, embedded subprocesses, event subprocesses, call activities, parallel and sequential multi-instance activities. Additionally, it provides advanced event handling with support to interrupting and non-interrupting events of type message, error, escalation, signal and terminate, as per the standard semantics.

4 About the Demo

For the demo, we will use the simplified model of a loan assessment process shown in Fig. 3. To make the models executable, some code needs to be added, including: process variable definitions, solidity code for script tasks, and code specifying the exchange of information from/to user/service tasks. For example, one must specify the information an applicant must provide via the task Enter loan information. This is done by writing the following snippet in the BPMN documentation element under this task:

```
(uint _monthlyRevenue, uint _loanAmount, uint _propertyReferenceNumber) : (uint montlyRevenue,
    uint loanAmount, uint propertyReferenceNumber) -> {montlyRevenue = _montlyRevenue;
      loanAmount = _loanAmount; propertyReferenceNumber = _propertyReferenceNumber; }
```

This snippet specifies that an applicant must provide the loan information, e.g. via a web form as shown in Fig. 2. The second part of the snippet is used to generate code that copies the information into process variables. Similar snippets are associated to the remaining tasks of the process model.

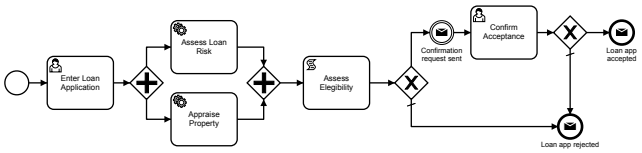

Fig. 3. Simplified loan assessment process model

5 Source Code and Screencast

The source code of Caterpillar can be downloaded under the BSD 3-clause "New" or "Revised" License from `https://github.com/orlenyslp/Caterpillar`.[5]

Caterpillar's code distribution contains three folders: (i) folder *caterpillar_core*, which includes the implementation of the core components (engine, compilation tools, event monitor and work item manager); (ii) *execution_panel* (implementation of the Execution Panel module); and (iii) *services*, which contains the implementation of the external services used in the demonstration. The repository contains all instructions needed to install the required dependencies and running the sample process model.

A screencast of Caterpillar can be found at `https://youtu.be/zOmx5kD1Og0` .

References

1. Jan Mendling et al. Blockchains for business process management - challenges and opportunities. *CoRR*, abs/1704.03610, 2017.
2. L. García-Bañuelos, A. Ponomarev, M. Dumas, and I. Weber. Optimized Execution of Business Processes on Blockchain. In *BPM 2017*, LNCS. Springer, 2017.
3. Object Management Group. Business Process Model and Notation, version 2.0. http://www.omg.org/spec/BPMN/2.0/.
4. I. Weber, X. Xu, R. Riveret, G. Governatori, A. Ponomarev, and J. Mendling. Untrusted Business Process Monitoring and Execution Using Blockchain. In *BPM 2016*, LNCS 9850, pages 329–347. Springer, 2016.

[5] The fourth author was not involved in the development or release of the Caterpillar software. Neither the publication of this paper nor the software release should be construed as granting any rights in relation to patents or patent applications held by CSIRO or by the authors of this paper.