Blockchain Support for Collaborative Business Processes

Claudio Di Ciccio · Alessio Cecconi
Marlon Dumas
Luciano García-Bañuelos
Orlenys López-Pintado · Qinghua Lu
Jan Mendling · Alexander Ponomarev
An Binh Tran · Ingo Weber

Introduction

Collaboration between different organizations is essential to achieve greater, common goals. Consider for example a supply chain, where collaboration of different companies yields a product via the different steps from production to delivery [11]. As of today, the integration of processes of each involved party requires extensive information exchange. This makes the design and management of such interorganizational business processes difficult. Furthermore, the multitude of message exchanges entails data redundancy and lack of full knowledge of how, when and where tasks have been conducted. For these reasons, companies still rely on authorized third parties to mediate and control the execution of interorganizational business processes.

Blockchain technology offers the unprecedented capability to support such processes [10]. The blockchain as a totally ordered data structure can capture the history and the current state of the business processes, whose transitions are registered by the transactions. As it is tamper-proof, the logging of executed processes cannot be subject to dispute on counterfeiting actions from process actors or third parties. Since it is replicated among the nodes in the network, the information on the process state can be shared and updated locally to every node, thus allowing the process participants to monitor the new process transitions and, if necessary, be readily prompted to the next action. Other interested parties such as auditors can inspect past executions for compliance. The pseudonymity guaranteed by the protocol enables collaboration in open environments. The programmability offered by blockchains such as Ethereum [2] is paramount to implement the workflow, as smart contracts can encode the business logic of processes and enforce its rules by design. Consensus algorithms, thus, create a trustworthy infrastructure on top of potentially untrusted nodes, and smart contracts make for a trusted process execution among partially trustable parties. As a result, although the technology is still novel, its adoption as a backbone for business process management systems (BPMSs) is rapidly evolving towards a general approach for executing business processes.

In this paper we illustrate how to design and run interorganizational business processes using blockchains. In particular, we illustrate the principles and rationale of the model-driven approach to business process automation on blockchains and then report on recent advances in the field.

The remainder of the paper is structured as follows. In Section "Model-Driven Engineering and Execution of Blockchain Processes", we provide the motivation for our work and provide an overview of

https://doi.org/10.1007/s00287-019-01178-x

Claudio Di Ciccio · Alessio Cecconi · Jan Mendling
Wirtschaftsuniversität Wien,
Vienna, Austria
E-Mail: {claudio.di.ciccio, alessio.cecconi, jan.mendling}@wu.ac.at

Marlon Dumas · Luciano García-Bañuelos · Orlenys López-Pintado
University of Tartu,
Tartu, Estonia
E-Mail: {marlon.dumas, luciano.garcia, orlenyslp}@ut.ee

Qinghua Lu · Alexander Ponomarev · An Binh Tran · Ingo Weber
Data61, CSIRO,
Sydney, NSW, Australia
E-Mail: {qinghua.lu, alexander.ponomarev, an.binh.tran, ingo.weber}@data61.csiro.au

Abstract
Blockchain technology provides basic building blocks to support the execution of collaborative business processes involving mutually untrusted parties in a decentralized environment. Several research proposals have demonstrated the feasibility of designing blockchain-based collaborative business processes using a high-level notation, such as the Business Process Model and Notation (BPMN), and thereon automatically generating the code artifacts required to execute these processes on a blockchain platform. In this paper, we present the conceptual foundations of model-driven approaches for blockchain-based collaborative process execution and we compare two concrete approaches, namely Caterpillar and Lorikeet.

how to design interorganizational business processes for execution on the blockchain. Thereupon, we describe the support of process execution as provided by the novel Lorikeet [12] (Section "The Lorikeet System") and Caterpillar [8] (Section "The Caterpillar System") systems. Finally, in Section "Discussion and Outlook", we discuss the current status and look at the future of this research stream.

Model-Driven Engineering and Execution of Blockchain Processes

Blockchains can operate as decentralized programmable platforms [15]. Platforms such as Ethereum [2] allow for the encoding of *smart contracts*, namely fully executable distributed programs operating on the blockchain. Smart contracts dictate how business is to be conducted among contracting parties. They, thus, naturally allow for the enactment of processes on-chain. The knowledge of programming languages for blockchain-specific applications is crucial to understand the behavior of those programs, which tend to become monolithic algorithms regulating the data to exchange, the control-flow logic, the access grants and the business rules.

Working in the blockchain is mostly a prerogative of individuals with such technical knowledge. Furthermore, existing solutions for business-to-blockchain services are often tailored to customers on a case-based approach. Organizations willing to move the coordination and tracking of the information exchange of their inter-organizational business processes onto the blockchain are thus hampered by technical skills requirement. Managers and business analysts should be interfaced through a model-driven approach, thus reducing the need to know encoding language details to create, manage and verify smart contracts underpinning the collaborative processes [16, Ch. 1].

In traditional business process management, this abstraction has long been established through processes notations like BPMN [4]. A modern BPMS lets the user define and manage a process through high-level notation. The system then manages the implementation details, thus hiding the streamline between the high-level design language and the software code behind the executable process. At the time of writing, such an abstraction has not yet been achieved for the blockchain. With blockchains, the problem is twofold: on the one hand, there is the need for abstracting the design from the implementation of smart contracts; on the other hand, a framework for their integration with processes is yet to be defined. The preference is to not introduce yet another process modeling language but to rely on existing established ones extended with blockchain features.

The design of smart contracts should be comprehensible, fast, reliable and verifiable. The automated generation of smart contracts code as interfaces to business processes can allow for faster prototyping and inspection already at the level of models. This would diminish the hindrance to the alignment between the expected behavior of the business process, and its implementation as supported by the blockchain. This solution would bring a higher degree of customizability than a catalogue of contract templates, such as the ones offered by the Corda blockchain.

To answer that call for standardization, different proposals are emerging from academia and industry. These proposals attempt to apply already existing interorganizational process representations, including process choreographies [9, 14] and orchestration diagrams [8] from the BPMN standard.

Figure 1 depicts a choreography diagram, namely a particular type of process representation focused on the information flow between organizations, rather than on the internal workflows of each actor. The parties in a collaborative business

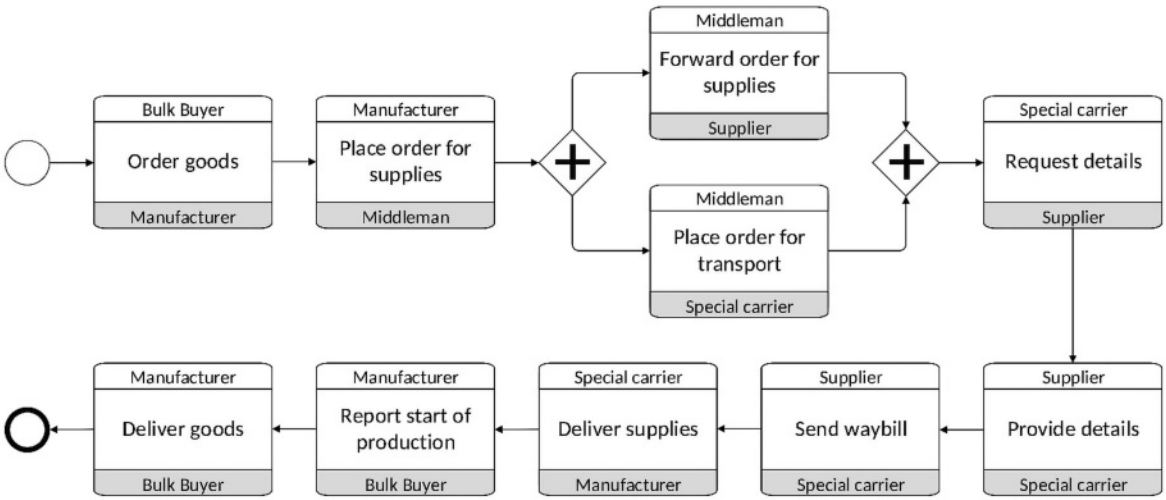

Fig. 1 A process choreography diagram

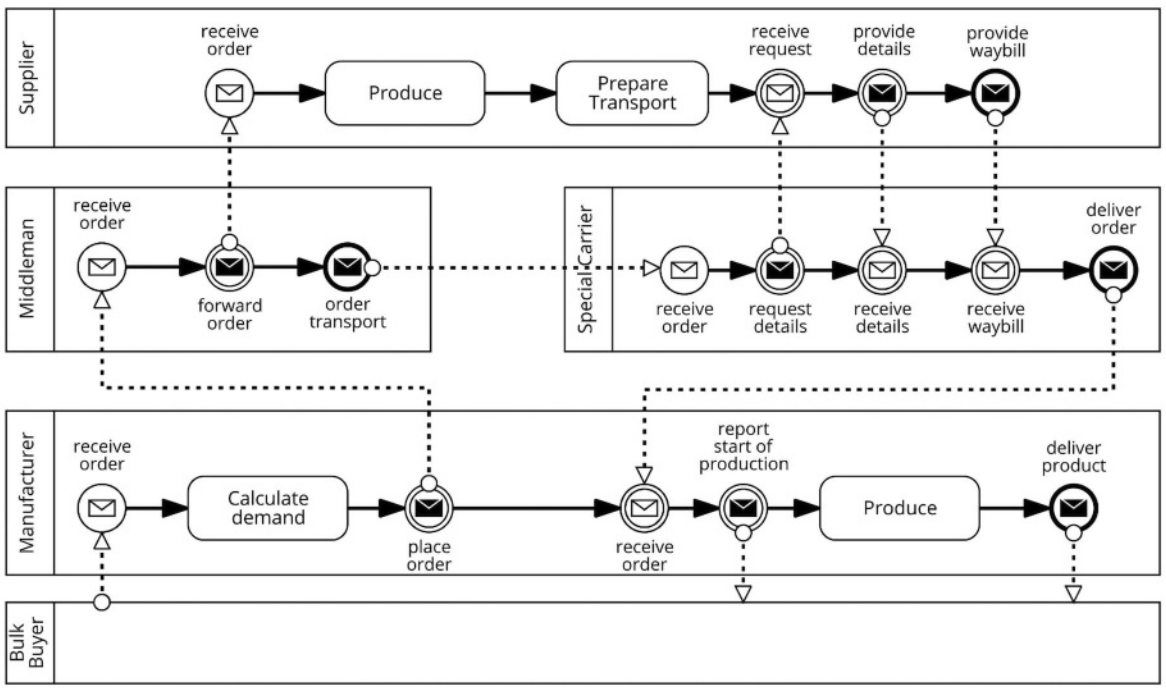

Fig. 2 The process in Fig. 1 as a collaboration diagram [14]

process cooperate through message exchanges. The blockchain records these messages and checks or enforces that these exchanges occur in a certain order. The process of each party remains off-chain and is hidden from the other actors.

An orchestration diagram such as the one in Fig. 2 details the individual processes of the participants. Communications are depicted as in-bound or out-bound message exchanges occurring when certain activities are performed. The blockchain is the shared process execution platform where every party executes their sub-processes and send or receive messages within the orchestration.

In the following sections, we present two novel examples of systems enacting processes on the blockchain: Lorikeet and Caterpillar. Both sys-

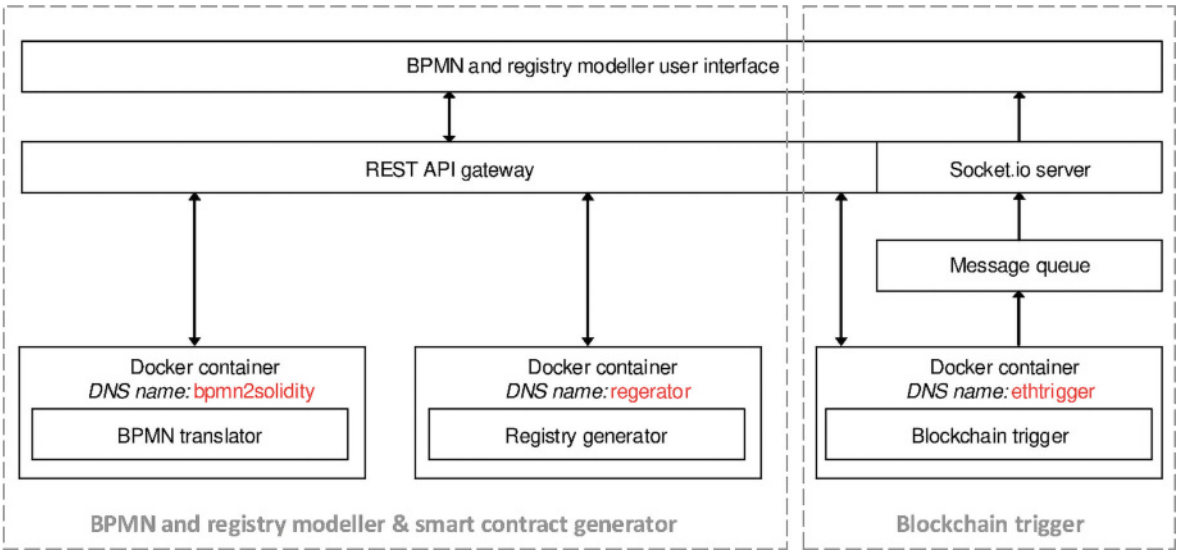

Fig. 3 The software architecture of Lorikeet

tems are based on the Ethereum blockchain and allow for smart contract deployment on public, private or permissioned blockchains. The former uses the blockchain as the message exchange mechanism for process choreographies, while the latter deploys the entire collaborative process on-chain.

The Lorikeet System

Overview and Design Principles

Lorikeet is a model-driven engineering (MDE) tool for the development of blockchain applications in the space of business processes and asset control. Management of assets is considered to be the first "killer application" of the blockchain, starting with fungible assets like cryptocurrency (Bitcoin, Ether, etc.) and tokens (second-tier coins that are managed on existing blockchain networks like Ethereum, such as Golem or Gnosis). Business processes that manage nonfungible assets (e. g., by transferring cars or land titles) are a promising application domain for blockchains. Unlike fungible assets, nonfungible assets can be highly individualized, which can introduce inefficiencies and uncertainty, leading to counterparty risks. The management of nonfungible assets traditionally relies on a centralized trusted authority, which again causes trust issues. A system that enables the automation of business processes on the blockchain, therefore, should not overlook the aspects pertaining to asset management.

Lorikeet can automatically produce blockchain smart contracts from business process models and asset data schemata. Lorikeet incorporates the registry editor Regerator [13] and implements the BPMN translation algorithms from both [14] and [5]. In a traditional BPMS, the process model is the artifact or blueprint which is enacted. In contrast, Lorikeet creates the code that implements the process, and the code is subsequently executed. The generated code can be reviewed, adapted or augmented before execution. This feature supports the potential need for building trust into the smart contracts generated and helps technical experts understand the code. Furthermore, it allows for rapid prototyping at the beginning and later extension toward a production system. Finally, MDE allows for amendment of the code beyond the expressiveness of process model notations. It is an entirely separate tool from Caterpillar [7], which we describe in the next section. Lorikeet is in commercial use by Data61 and has been applied in numerous industry projects. A demonstration paper [12] outlines the tool architecture and usage. The content of this section is partly based on that publication.

Architecture

Lorikeet is a well-evaluated tool that is used for creating blockchain smart contracts in industry and academia. Figure 3 illustrates the architecture of Lorikeet, which consists of a modeler user interface

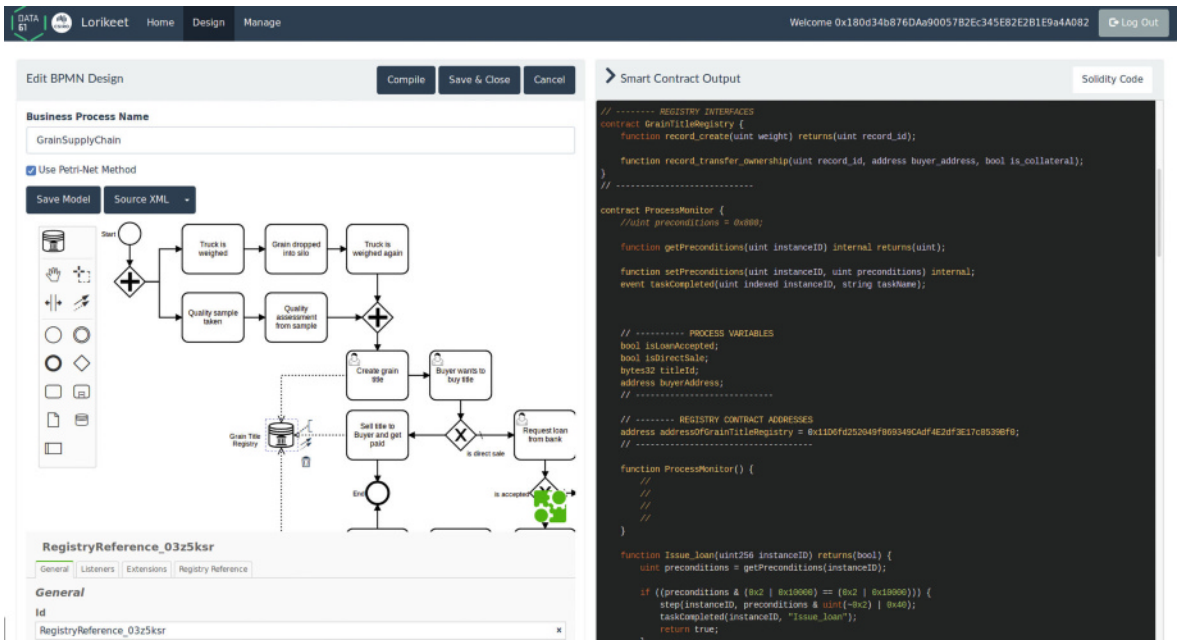

Fig. 4 A screenshot of the Lorikeet business process modeler

(UI, shown in Fig. 4), and back-end components including the BPMN translator, Registry generator and Blockchain trigger.

The modeler UI component is presented as a web application for users to build business process and registry models. Business processes are modeled in BPMN 2.0. Lorikeet extends that standard to support representation of, and interaction with, registries in the BPMN process model. The extension comprises two new elements, namely RegistryReference and ActionInvocation, in terms of new graphical notations and new XML attributes. A RegistryReference represents an asset data store on a blockchain, while an ActionInvocation shows the asset registry action to invoke. On the registry side, the registry modeler provides a form for users to fill in the registry model input. The registry model consists of four parts, including basic information (registry name, description, user-defined data fields and their types), registry type (single or distributed), basic CRUD operations and advanced operations (record lifecycle management and foreign key). Specifically, an access control policy is provided to regulate the registry manipulations. Process instances can also manipulate the registry records. For an action invocation from the process instance to the registry, changes to the registry record are finalized only after the execution of the process step logic is completed. Since registry actions could fail, there is a check box on Lorikeet's user interface for atomic behavior across registry actions and business process tasks: if marked as atomic, the registry update and the process state change either both fail or succeed.

The back-end components, including the *BPMN translator*, *Registry generator*, and *Blockchain trigger*, are built to adhere to a microservice-based architecture and are deployed independently as Docker containers. The *BPMN translator* automatically generates Solidity smart contracts from the aforementioned BPMN models. The smart contracts include the information to call registry functions and to instantiate and execute the process model. The *Registry generator* creates Solidity smart contracts as well, based on registry models that provide information on the data structure, registry types, plus basic and advanced operations. Users can then deploy the smart contracts on the blockchain. The *Blockchain trigger* communicates with an Ethereum blockchain node and handles compilation, deployment and interaction with Solidity smart contracts.

The *BPMN and registry modeler UI* interacts with the back-end microservices via an API gateway. The API gateway forwards API calls from the modeler UI, such as translating a BPMN model,

to the corresponding microservice. Figure 4 depicts the business process modeler UI of Lorikeet. It is split into two panels: one for modeling processes in BPMN, on the left-hand side, and one showing the source code, on the right. Once the user applies changes to the BPMN model, the corresponding Solidity smart contract code is altered correspondingly at design time. The appearance and concept of the registry modeler UI is similar to the business process modeler UI. Lorikeet has been used within international collaborations with academics and industry partners. A screencast demonstrating the usage of the Lorikeet tool can be found at https://drive.google.com/open?id=1rpy-oHbDVkXa6u4Fn73wSX8rINn1sv3U.

The Caterpillar System

Overview and Design Principles

Caterpillar's aim is to enable its users to build native blockchain applications to enforce the correct execution of collaborative business processes starting from a BPMN process model. In this context, the meaning of "native" is that code artifacts deployed on the blockchain encode all the execution logic captured in the process model. Specifically, Caterpillar aims at fulfilling the following three design principles [8]. First, the collaborative process is modeled as if all the parties shared the same process execution infrastructure (the blockchain). Accordingly, the starting point for implementing a collaborative business process is a single-pool BPMN process model and not a collaborative process model or a choreography model where parties communicate via message exchanges. Second, the full state of the process instance and of its subprocess instances is recorded on the blockchain. All the metadata required in order to retrieve the links between a given process instance and its related subprocess instances are also recorded on the blockchain. Third, the execution of a process instance can proceed even if all the off-chain components of Caterpillar are unavailable.

To achieve these principles, Caterpillar translates a BPMN process model into a set of smart contracts, which can enforce the business process without making assumptions about any of the off-chain components that trigger transactions on these contracts. While Caterpillar comes with off-chain components for deploying, triggering and monitoring business processes, these off-chain components are optional. Parties can, thus, either directly invoke the smart contract transactions without going through Caterpillar's off-chain components or implement their own runtime. Accordingly, multiple instances of the off-chain runtime component may be running simultaneously (e. g., one instance per participant).

Architecture

The architecture of Caterpillar comprises three layers, as shown in Fig. 5. The lower layer, namely the *On-Chain Runtime*, implements the process execution logic as a set of smart contracts spread across five components. The central component of this layer, namely the *Workflow Handler*, encodes the control-flow perspective of the process. This component is responsible for determining which tasks (work items) are enabled within a given process instance. On the right-hand side, the *Service Bridge* and *Worklist Handler* manage the interactions with external applications and users and validate the data produced by the execution of a task. The fourth component, *Contract Factories*, provides a configuration mechanism to create new instances of a process. For example, a *Factory* establishes which worklist is used by the users to interact with the process or how a process must be bound to a subprocess in the process hierarchy. Finally, the *Runtime Registry* is a smart contract that keeps track of the process instances and their relation with other contracts in the *On-Chain Runtime*. The *Runtime Registry* is a critical component of the architecture, as it allows for the recovery of the status of any process instance, independently of any application monitoring the process off-chain. The *Ethereum Log* provides a medium for interaction between off-chain and on-chain components. For example, as the execution of a task must be mined as a transaction in the blockchain, an event can be emitted in the log to notify the external components that the miners accepted the execution. Finally, the *Process Repository* stores and provides access to compilation artifacts and metadata required to deploy the smart contracts and tracking the process instances off-chain. Unlike the *On-chain Runtime* and the *Ethereum Log*, the *Process Repository* is stored off-chain in other decentralized networks like the InterPlanetary File System (IPFS), which provides

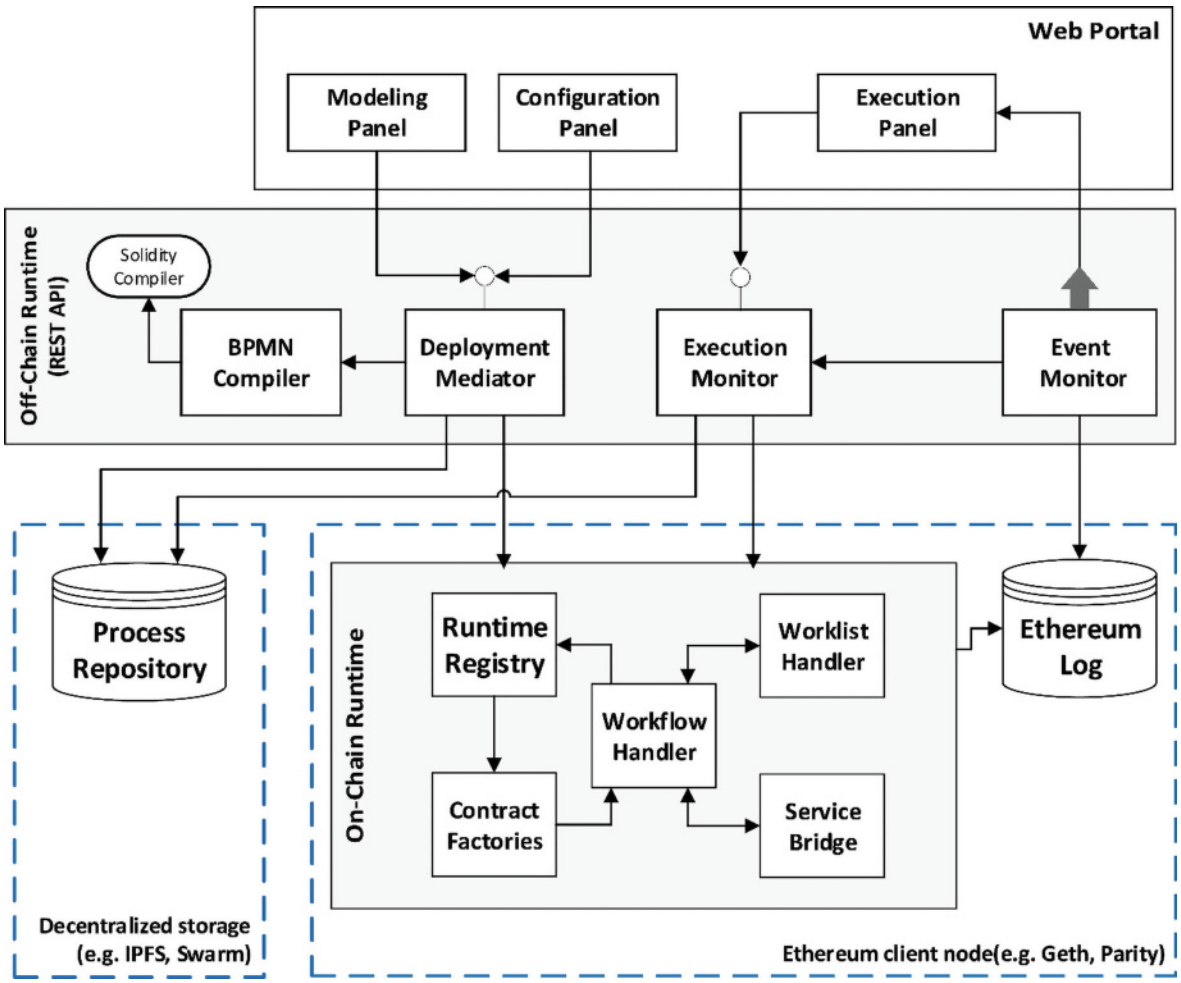

Fig. 5 The software architecture of Caterpillar

tamper-resilient storage but at a lower cost than the Ethereum blockchain.

The middle layer, namely the *Off-Chain Runtime*, provides a means for external applications to interact with the components of the bottom layer in a service-oriented fashion. The *Off-Chain Runtime* consists of five components. From left to right, the first one is the *BPMN Compiler*, which is responsible for translating the BPMN models into smart contracts. In a second step, the *BPMN Compiler* interacts with a standard *Solidity Compiler* to produce the metadata (i. e., EVM bytecode and ABI definitions) required for the deployment of the smart contracts in Ethereum. The *Deployment Mediator* is responsible for creating new instances of the process. In addition, the *Deployment Mediator* triggers the compilation of a model and serves to (re)bind process contracts, factories, worklists and services, not necessarily produced by Caterpillar but relying on the structure outlined by its interfaces. The *Execution Monitor* component allows for the querying of the process execution state (e. g., which tasks are enabled to be executed and to execute such tasks). Last, the *Event Monitor* is a listener component for events emitted in the *Ethereum Log*, which pushes notifications to the *Execution Monitor* or other external applications.

The top layer, namely the *Web Portal*, exposes the functionalities of the *Off-Chain Runtime* components to end users (e. g., the process stakeholders). The *Web Portal* provides three panels. First, the *Modeling Panel* allows drawing or uploading BPMN models which are deployed later via the *Deployment Mediator*. Second, the *Configuration Panel* supports the binding/rebinding of relations on the process contracts already deployed. Last, the *Execution Panel*, depicted in Fig. 6, offers a visual representa-

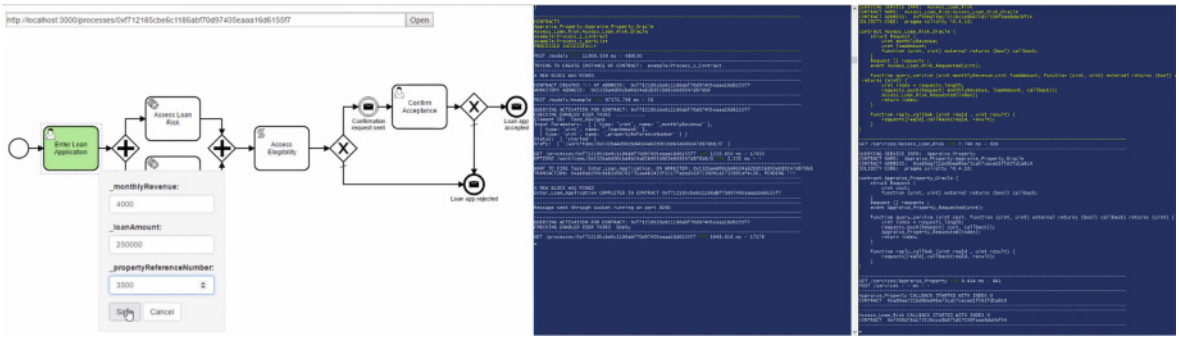

Fig. 6 A screenshot of the Caterpillar execution panel

tion of the process status and allows for the execution of tasks.

Discussion and Outlook

In this paper, we analyzed the current state of the art for the model-driven design and implementation of blockchain-based process execution and monitoring – the first research challenge in [10]. In order to ground the analysis, we presented two research prototypes in the field, namely Caterpillar and Lorikeet. Table 1 compares the features of these two systems. They share the common aim of exploiting the features of blockchain technology to ensure that a collaborative business process, involving untrusted parties, abides to a given BPMN process model. Moreover, they both rely on compiling a BPMN process model into Solidity smart contracts, with the difference that Caterpillar additionally provides predefined runtime components that are packaged together with the compiled code. Lorikeet supports asset control and access control to restrict access to operations and data, whereas Caterpillar, in its current version, is focused on control-flow aspects. On the other hand, Caterpillar provides almost full coverage of the control-flow perspective of BPMN, whereas Lorikeet provides limited support.

Both Lorikeet and Caterpillar are focused on BPMN-style process models. Other proposals have investigated the possibility of modeling blockchain-based collaborative process models using alternative paradigms such as artifact-centric process modeling [6]. The advancements brought about by those endeavors have already facilitated applications such as the automated tracking of process instances on the blockchain [3] and will aid more advanced tasks, like automated discovery and auditing for blockchain-based processes.

Important issues to resolve for future studies mainly pertain to the connection between real-world objects and the digital space of blockchains. Indeed, business processes often have a tight bond

Table 1

Features of Lorikeet and Caterpillar

	Lorikeet	Caterpillar
Model execution approach	MDE (code generation)	MDE (code generation)
BPMN elements support	Medium	High
Discovery of incorrect behavior	Supported	Supported
Sequence enforcement	Supported	Supported
Participant selection	Predefined	N/A
Data sharing	Mixed	All
Asset control	Supported	Not supported
Access control	Supported	Not supported

with physical assets and resources. Therefore, updates on their status should be notified within the blockchain space, and commands triggered from smart contracts may need to be conveyed outwards to actually modify their status. To that extent, the role of in-bound and out-bound oracles seem prominent. Studies on the requirements that they have to meet in this context and investigations on their integration with existing approaches, are, therefore, paramount. Another aspect to be examined is that transactions recorded in the most recent blocks cannot be taken for granted, as blockchains guarantee only eventual consistency as long as Proof of Work is the main consensus model. This fact collides with the expenses or hurdles that an off-chain rollback or compensation might require, should commands be executed on the physical world based on transactions from forks that are no longer valid. By the same line of reasoning, engineering the data management is key in the context of process automation, and even more so when blockchains come into play. The trade-off between on-chain and off-chain data involves both governance and pragmatic aspects. On the one hand, the storage of process data on-chain entails higher gas costs and poses security threats. On the other hand, resorting to off-chain data signifies the potential renouncement of tamper-proof, signed recordings, and looser links between process flows and data flows. Either way, studies should be conducted to address by design the privacy concerns that arise when sensitive data are stored in the context of process execution [1].

As outlined in [10], the research communities involved will engage in endeavors in diverse directions, including not only the aforementioned technical ones, but also regarding the strategic decisions of the organizations embracing the blockchain technology to implement their interorganizational processes. With this paper, by proposing nascent conceptual foundations of model-driven approaches for blockchain-based collaborative process execution, we have reported on the initial, promising steps toward overcoming the many challenges that lie ahead of scientific investigators.

Funding. Open access funding provided by Vienna University of Economics and Business (WU).

References

1. Basin D, Debois S, Hildebrandt T (2018) On purpose and by necessity: compliance under the GDPR. In: FC. Springer, Berlin Heidelberg
2. Dannen C (2017) Introducing Ethereum and Solidity: Foundations of Cryptocurrency and Blockchain Programming for Beginners. Apress
3. Di Ciccio C, Cecconi A, Mendling J, Felix D, Haas D, Lilek D, Riel F, Rumpl A, Uhlig P (2018) Blockchain-based traceability of inter-organisational business processes. In: BMSD. Springer, Cham, pp 56–68
4. Dumas M, La Rosa M, Mendling J, Reijers HA (2018) Fundamentals of business process management. 2nd edn. Springer, Berlin Heidelberg
5. García-Bañuelos L, Ponomarev A, Dumas M, Weber I (2017) Optimized execution of business processes on blockchain. In: BPM. Springer, Cham, pp 130–146
6. Hull R, Batra VS, Chen Y, Deutsch A, Heath III FFT, Vianu V (2016) Towards a shared ledger business collaboration language based on data-aware processes. In: ICSOC. Springer, Cham, pp 18–36
7. López-Pintado O, García-Bañuelos L, Dumas M, Weber I (2017) Caterpillar: A blockchain-based business process management system. In: BPM Demos. CEUR-WS
8. López-Pintado O, García-Bañuelos L, Dumas M, Weber I, Ponomarev A (2018) CATERPILLAR: A business process execution engine on the Ethereum blockchain. Technical report. http://arxiv.org/abs/1808.03517
9. Madsen MF, Gaub M, Høgnason T, Kirkbro ME, Slaats T, Debois S (2018) Collaboration among adversaries: Distributed workflow execution on a blockchain. In: FAB
10. Mendling J, Weber I, van der Aalst WMP, vom Brocke J, Cabanillas C, Daniel F, Debois S, Di Ciccio C, Dumas M, Dustdar S, Gal A, García-Bañuelos L, Governatori G, Hull R, La Rosa M, Leopold H, Leymann F, Recker J, Reichert M, Reijers HA, Rinderle-Ma S, Solti A, Rosemann M, Schulte S, Singh MP, Slaats T, Staples M, Weber B, Weidlich M, Weske M, Xu X, Zhu L (2018) Blockchains for business process management – challenges and opportunities. ACM Trans Manag Inf Syst 9(1):4:1–4:16
11. Snyder LV, Shen ZJM (2011) Fundamentals of supply chain theory. Wiley, Hoboken
12. Tran AB, Lu Q, Weber I (2018) Lorikeet: A model-driven engineering tool for blockchain-based business process execution and asset management. In: BPM Demos. CEUR-WS, pp 56–60
13. Tran AB, Xu X, Weber I, Staples M, Rimba P (2017) Regerator: a registry generator for blockchain. In: CAiSE Forum. CEUR-WS, pp 81–88
14. Weber I, Xu X, Riveret R, Governatori G, Ponomarev A, Mendling J (2016) Untrusted business process monitoring and execution using blockchain. In: BPM. Springer, Cham, pp 329–347
15. Wood G (2017) Ethereum: A secure decentralised generalised transaction ledger eip-150 revision (759dccd – 2017-08-07). https://ethereum.github.io/yellowpaper/paper.pdf. Accessed: 2018-01-03
16. Xu X, Weber I, Staples M (2019) Architecture for blockchain applications. Springer, Berlin Heidelberg