Business Process Execution on Blockchain

Orlenys López-Pintado
Supervisors: Luciano García-Bañuelos and Marlon Dumas

University of Tartu, Estonia
orlenyslp@ut.ee

Abstract. Nowadays, in traditional Business Process Management Systems (BPMSs) supporting inter-organizational collaborations issues like the lack of trust and the flexibility to adapt to changes quickly are still a roadblock. Blockchain technology provides a tamper-proof mechanism for decentralized execution of collaborative business processes. In that context, a blockchain can serve not only to keep track of the exchanged information but also to coordinate, via smart contracts, the execution of steps as prescribed by a process model. Nevertheless, current solutions in this field are still in an early stage or at a high level of abstraction. This research aims at developing a BPMS on blockchain to handle inter-organizational collaborations that relies on the translation of process models captured in the Business Process Model and Notation (BPMN) into smart contracts. Unlike existing solutions, we consider the blockchain as the platform supporting the full execution of a process. This work presents the main challenges to be tackled, outlines the research approach to follow and describes some preliminary results on designing the architecture of the prototype and compiling process models into smart contracts.

Keywords: Business Process Management Systems, Blockchain, Business process execution, Smart contracts

1 Introduction

Business Process Management Systems (BPMS) are information systems that automate the execution of business processes typically described by a model. They coordinate the interactions of human actors with machine operations providing to the organizations a flexible way to manage their processes. Nowadays, a big variety of BPMSs cover some of the traditional steps of the business process lifecycle, i.e. identification, discovery, analysis, redesign, implementation, execution, monitoring, adaptation and evolution [4]. However, typical issues like interoperability, dynamic interactions, trust and security are not fully addressed in inter-organizational collaborations between mutually untrusted parties [16].

Several challenges exist in the field of inter-organizational collaborations. One of them is the autonomy of organizations and the lack of trust, e.g. which partner performs a given task, what data should be accessed and by who, and so forth. Although some agreements can be reached, choosing a trustworthy mechanism

or a middleman can be a problem even bigger than the process automation itself [16]. Another challenge is flexibility to allow changes on the process requirements in real time [15, 17], such as dynamic onboarding and replacement of business parties. In the logistics field, some common collaborative processes require dynamic binding and re-binding. For example, in a buyer-supplier-carrier process, the carrier might sometimes be appointed (selected) by the supplier, and other times by the buyer. Also sometimes the seller might have the right to change the carrier after the initial appointment, e.g. if the initially appointed carrier is not able to pick up the merchandise on time. In other words, the responsibility for a given task can change across cases.

Blockchain provides a suitable platform to execute inter-organizational processes in a trusted manner [8]. The above is supported by two main concepts of the blockchain technology. First, no central authority is required but a peer-to-peer network store a copy of the information as a linked sequence of blocks. Thus, each block contains a set of valid transactions. Second, the inclusion of smart contracts provides support to implement businesses rules and updating the process status as programmable scripts [1].

The use of blockchain as a distributed ledger to handle different collaborative scenarios has been investigated in [19]. By executing inter-organizational process as smart contracts on blockchain some common issues can be solved. First, blockchain will serve as an immutable and trusted ledger that records and validates every operation during the process execution. Transactions can be restricted to a set of roles, i.e. they can be executed only by a participant with an authorized signature. In addition, other business rules can be implemented by the smart contracts to perform the tasks. Participants may access the transactions generated by the process which enable the monitoring of its execution. Besides, with encryption, it is possible to restrict the access to data that is public but only readable by participants with the corresponding privileges [11].

Implementing business processes using the low-level primitives provided by blockchain platforms is cumbersome. In contrast, established Business Process Management Systems (BPMSs) provide convenient abstractions for rapid development of process-oriented applications. Aligned with that vision, the problem to address in this research project is to design a BPMS on blockchain to handle inter-organizational collaborations.

2 Related Work

The first work addressing the problem of lack of trust on inter-organizational business processes using blockchain was reported in [20]. The authors introduce the architecture of an Ethereum-based platform for tracking the execution of collaborative processes specified as choreographies in Business Process Model Notation(BPMN) [13]. In a BPMN choreography, the aim is to capture the exchange of messages among the partners that can be cumbersome. Ultimately, it is a weak approach to build consensus which makes the process execution less

flexible (e.g. dealing with dynamic on-boarding or rebinding of parties requires more message exchanges).

In [6] the authors explore approaches to transform BPMN models into Solidity smart contracts. Additionally, some strategies to reduce the cost of deployment and execution of the smart contracts are presented. However, advanced process model constructions such as embedded and external subprocesses, multi-instances activities and all kind of event propagation are not covered.

In [9] is discussed a vision on how the artifact-centric paradigm [3] can be suitable to model business collaborations over blockchain technology. Similarly, the author of [12] advocates the use of blockchain to coordinate collaborative business processes based on choreography models. Finally, [5] presents a high-level solution to map smart contracts from domain-specific languages. However, all these efforts are still nascent and we are not aware of any concrete implementation of the ideas presented in the three papers above.

From the BPM industry, we can mention an open source BPMS owned by a consulting company named Bonitasoft [14]. The system allows integrating a BPMN process model with the blockchain platform offered by Chain Inc. The solution provides a software connector which enables process instances to forward requests to the blockchain. However, the approach is different from our vision because we consider the process instance running on the blockchain

3 Design Principles and Research Questions

The closest work to our problem statement was presented in [20]. There, collaborative business processes are captured via BPMN choreographies which are a weak approach to build consensus as described in Section 2. The authors consider the blockchain as a storage for tracking messages exchanged by external business processes running on conventional BPMSs. Conversely, we see the blockchain technology as the platform supporting the execution of a full-fledged BPMS. Consequently, the underlying architecture of such system is radically different.

We aim to develop a prototype which will provide high-level of abstraction based on the following principles: (i) The process must be modeled as a single-pool model in BPMN [13], i.e not a collaborative process nor a choreography. Each party is represented by a lane. Interactions between two parties are modeled by sequence flows that go from one lane to another. (ii) Process state is stored in the blockchain. (iii) The execution logic of the process model is translated into smart contracts which must run independently of any other off-chain component. In other words, the execution of a process instance runs autonomously in the blockchain, no matter if it was deployed by an external application that is interrupted later. Accordingly, the research will be structured to address the following research questions:

1. How to compile the standard of BPMN to execute processes as smart contracts?

1.1. How to compile simple BPMN models including only tasks and gateways?
1.2. How to compile BPMN models with subprocesses and multi-instances?
1.3. How to compile BPMN models to allow event handling?
1.4. How to compile BPMN models including data management?
1.5. How to compile BPMN models with tasks involving interaction with external resources (e.g., user and service)?

2. What would be a suitable software architecture to develop a business process management system on top of blockchain?
2.1. How feasible is it to use traditional BPMSs architectures on blockchain platforms?
2.2. How to develop de the system through off-chain and on-chain operations?
2.3. How to store the data required by the process execution?
2.4. How to guaranty dynamism, flexibility and autonomy of processes deployed therein?

3. How to implement an access control mechanism suitable for inter-organizational processes between mutually untrusted parties?
3.1. What would access control mechanisms allow us to capture the wide range of dynamic binding and rebinding scenarios found in collaborative business processes?
3.2. How can these access control mechanisms be seamlessly integrated into the standard BPMN notation?
3.3. How can the resulting access control-enhanced BPMN models be compiled into smart contracts?
3.4. How to protect/validate data received exchanged among untrusted resources?

4. How to optimize the code generated by the prototype to improve the degree of concurrency to allow the deployment of large process models?

4 Research Methodology

To solve the problem, we are following the methodology of design science in information systems. The aim is to answer the research questions and implement a prototype iterating incrementally in cycles: *build-evaluate* [7]. We decompose every research question into multiple sub-questions such that every iteration answers one sub-question (sub-problem). Additionally, an iteration must follow the process: (i) check awareness of the problem to propose an initial solution, (ii) suggest a tentative design, (iii) develop an artifact and extend/update the prototype, (iv) evaluate the solution, (v) analyze the results to decide if move to the next iteration or repeat the process again in the current iteration to improve the result.

The evaluation follows the experimental approach described in [7]. In each iteration, the prototype is studied in a controlled environment to assess results. Synthetic and real-life datasets are used to simulate the process execution under different constraints. This experimentation includes coverage of the BPMN standard, interactions with stakeholders and analysis of system performance.

5 Preliminary Results

At the current stage, we have addressed the first two research questions. In addition, the architecture of the system supports a default access control policy that will be improved in next iterations. Results are implemented and tested in an open source prototype named Caterpillar [10]. The source code of the prototype may be accessed from `https://github.com/orlenyslp/Caterpillar`. Like most BPMSs, Caterpillar supports the creation of instances of process models in the standard BPMN 2.0. Moreover, the participants can track the state of process instances and execute tasks thereof.

The prototype is implemented on top of Ethereum [21]. The process models are compiled into Solidity[1], a contract-oriented programming language for Ethereum. In the following, we will describe briefly key aspects of Caterpillar.

5.1 Compiling BPMN to Solidity

As described before, Caterpillar assumes the input to be a BPMN process model and not a choreography. The current version of the Caterpillar compiler accepts models with tasks (default, user, script, service and receive), gateways (exclusive, parallel and event-based), events (default, terminate, message, signal, error and escalation), call-activities, embedded subprocesses, event-subprocesses and multi-instance activities (parallel and sequential) [13].

In most of the cases, Caterpillar produces at least two smart contracts from a flatten BPMN model. One to implement the control flow perspective. The second, named worklist, to handle the execution of workitems by stakeholders. In order to specify the data perspective of a process and how to manage it, the BPMN models are annotated with solidity snippets which are embedded later as fully executable instructions in the control flow and worklist contracts. These snippets can be global variables of the process, conditions for exclusive gateways, constraints to manage the data provided by process participants through user tasks, operations to perform by script tasks and so on.

The control flow perspective of a process is implemented by simulating a token game as specified in the BPMN standard [13]. In such game, a token is generated by the start event, which will traverse sequence flows in the model until an end event consumes it. Commonly, an activity is enabled, i.e. can be executed, if a token exists on any of its incoming edges.

Providing support to subprocesses and events poses a challenge. In such context, implementing the control flow perspective of a process model can require not a single contract (as in flattening models) but a hierarchy of smart contracts. Besides, a mechanism to propagate events and data along the hierarchy must be defined. For example, embedded subprocesses are designed inside a parent process. In this case, the steps required to execute the process, or to propagate data can be calculated statically in compilation time. On the other hand, call activities reference other contracts whose information may be unknown at compilation

[1] https://solidity.readthedocs.io/en/v0.4.21/

time. The above requires a mechanism to associate the smart contract encapsulating a subprocess linked to some activity. In addition, a common interface is required to propagate events between contracts probably defined from different viewpoints. Note that a process can be reused in distinct contexts. Lastly, event-subprocess are triggered by events and they are not part of the flow of its parent process. Consequently, they must be instantiated as result of catching an event propagated in the hierarchy.

The parsing of subprocesses and events also modifies the basic control flow approach used in flatten models. Here, checking the token distribution on the edges is not enough to determine if a process is running. For example, a child contract referenced by a call activity can be running and the parent is still active even without tokens on edges. In this case, tracking both token distribution and running activities (including transactions with pending validation in the blockchain) is required. On the other hand, boundary events and event-subprocesses do not fit in the typical token game because they have no incoming edge. Therefore, to check if they are enabled is required to check the element where they are attached or included depending on the case. In addition, a process must instantiate all its children and keep track when the execution is finished.

In order to handle events, Caterpillar provides three main types of propagation. One designed to send data from a child subprocess to its parent in the hierarchy. A second to propagate information from a parent to their children which can interrupt running instances if an error or abnormal termination occurs, e.g. when an interrupting boundary event is caught or a terminate end event is reached. Lastly, the third type is used to throw messages by publishing information in the event log of Ethereum[2]. All the event handling semantics presented in the BPMN standard [13] is fully covered for the six events (default, terminate, message, signal, error and escalation) allowed by the prototype.

5.2 System Architecture

The architecture of Caterpillar is presented in Fig. 1. The prototype distinguishes two types of components: (i) on-chain to represent operations implemented in the smart contracts running in the blockchain and (ii) off-chain referring the components interacting with the deployed contracts from outside of the blockchain.

The on-chain component named *Workflow Handler* manages the smart contracts compiled from the BPMN models, implementing the control-flow perspective. Besides, other two components named *Worklists Handler* and *Service Bridge* are aimed to handle the interactions with external stakeholders and information systems respectively. The contracts in these last two components must implement the access control mechanism of the related process. Although the compiler generates default contracts providing access to every participant, these can be replaced during the process execution.

[2] https://media.consensys.net/technical-introduction-to-events-and-logs-in-ethereum-a074d65dd61e

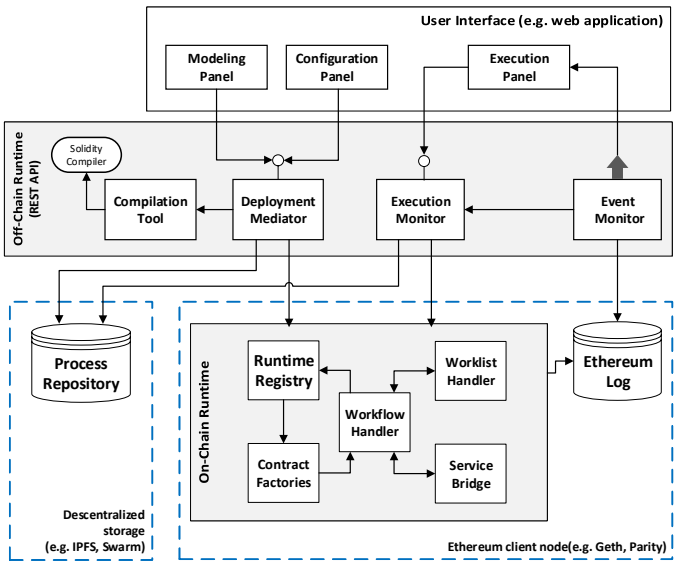

Fig. 1. Architecture of Caterpillar

The key component supporting the autonomy and flexibility of Caterpillar is named *Runtime Registry*. Relations among contracts are updated and accessed from the *registry*. On the other hand, *Contract Factories* are responsible to create new instances. For example, a factory establishes which worklist is related to a particular instance of a process. Note that the worklist can be different even among instances of the same contract. Indeed, the available factories are stored and accessed from the *registry*. Besides, the *Runtime Registry* keeps the addresses of any smart contract instantiated by the prototype as well as references to the compilation artifacts of the contracts. In other words, the history of process executions, active instances, relations between contracts and the metadata required to create new instances can be retrieved from the *Runtime Registry*.

Aligned with the idea to provide tamper-resilient storage, Caterpillar proposes the usage of a decentralized storage implemented on top of IPFS [2] to store the compilation metadata as well as information of the process required for its execution. IPFS-based storage produces a hash value that uniquely identifies each one of the compilation artifacts of a process model. Thus, this hash serves as a reference to recover the data required to create new instances of a process.

The functionalities enclosed in the off-chain components are exposed through a REST API that allows developers to interact with the on-chain components [10]. The API supports operations to compile and deploy the smart contacts from a BPMN model, to create instances of a contract, to configure relations/links in the registry, to find and execute enabled tasks and so on. An event monitor listens which events are stored in the log of the blockchain. Accordingly, push

notifications with information about the related transactions are generated. On top of the architecture, visual tools are provided in a web-based user interface to model, configure and execute process instances for usage of business analysts without any technical knowledge about the internal functioning of Caterpillar.

6 Outlook

The current version of Caterpillar lacks any access control mechanism, meaning that any participant can alter the state of execution of any process instance. The research question 3 aims to solve this gap. Mainstream BPMSs are largely based on a classical Role-Based Access Control Model. Concretely, each task in the process is mapped to a role. Any user (e.g. worker) who plays the role corresponding to a task can perform it. On top of this basic RBAC model, commercial BPMSs allow one to define constraints such as "retain familiar" (i.e. the same participant who performed a previous task should perform the current task) or "four-eyes principle" (the current task must not be performed by the participant who performed a previous task). Some BPMSs also support concepts of "delegation", as well as a wide range of resource allocation mechanisms [18]. Collaborative business processes, however, require more sophisticated access control mechanisms. In particular, some common collaborative processes in the field of logistics require dynamic binding and re-binding, as described in Section 1. Additionally, in some cases, an organization may require protecting its data from another untrusted party. In this situation, data encryption and policies to access it needs to be developed.

A limitation of the current prototype can be seen if the number of elements in the model is large. Here, the maximum amount of resources allowed per block (i.e. ether in the Ethereum platform) can be overflowed. However, divide and conquer approaches can be used while compiling to scale the process models. In addition, some strategies to optimize the code generated could be applied with the aim to reduce the costs of deployment and execution.

References

1. Bartoletti, M., Pompianu, L.: An empirical analysis of smart contracts: Platforms, applications, and design patterns. In: Financial Cryptography and Data Security - FC 2017 International Workshops, WAHC, BITCOIN, VOTING, WTSC, and TA, Sliema, Malta, April 7, 2017, pp. 494–509 (2017)
2. Benet, J.: IPFS - content addressed, versioned, P2P file system. CoRR **abs/1407.3561** (2014)
3. Cohn, D., Hull, R.: Business artifacts: A data-centric approach to modeling business operations and processes. IEEE Data Eng. Bull. **32**(3), 3–9 (2009)
4. Dumas, M., La Rosa, M., Mendling, J., Reijers, H.: Fundamentals of Business Process Management. Springer Berlin, Berlin (2018)
5. Frantz, C., Nowostawski, M.: From institutions to code: Towards automated generation of smart contracts. In: 2016 IEEE 1st International Workshops on Foundations and Applications of Self* Systems (FAS*W), Augsburg, Germany, September 12-16, 2016, pp. 210–215 (2016)

6. García-Bañuelos, L., Ponomarev, A., Dumas, M., Weber, I.: Optimized Execution of Business Processes on Blockchain. In: BPM 2017, LNCS. Springer (2017)
7. Hevner, A., March, S.T., Park, J., Ram, S.: Design science in information systems research. MIS Quarterly **28**(1), 75–105 (2004)
8. Hull, R.: Blockchain: Distributed event-based processing in a data-centric world: Extended abstract. In: Proceedings of the 11th ACM International Conference on Distributed and Event-based Systems, DEBS 2017, Barcelona, Spain, June 19-23, 2017, pp. 2–4 (2017)
9. Hull, R., Batra, V.S., Chen, Y.M., Deutsch, A., III, F.T.H., Vianu, V.: Towards a shared ledger business collaboration language based on data-aware processes. In: Service-Oriented Computing - 14th International Conference, ICSOC 2016, Banff, AB, Canada, October 10-13, 2016, Proceedings, pp. 18–36 (2016)
10. López-Pintado, O., García-Bañuelos, L., Dumas, M., Weber, I.: Caterpillar: A blockchain-based business process management system. In: Proceedings of the BPM Demo Track and BPM Dissertation Award co-located with 15th International Conference on Business Process Modeling (BPM 2017), Barcelona, Spain, September 13, 2017. (2017)
11. Mendling, J., et al.: Blockchains for business process management - challenges and opportunities. ACM Trans. Management Inf. Syst. **9**(1), 4:1–4:16 (2018)
12. Norta, A., Ma, L., Duan, Y., Rull, A., Kõlvart, M., Taveter, K.: eContractual choreography-language properties towards cross-organizational business collaboration. J. Internet Services and Applications **6**(1), 8:1–8:23 (2015)
13. Object Management Group: Business Process Model and Notation, version 2.0. URL `http://www.omg.org/spec/BPMN/2.0/`. Accessed 2018-04-08
14. Palacin, L.: Accelerate blockchain technology adoption with Bonita BPM and Chain Core. URL `https://vimeo.com/202058656`. Accessed 2018-04-08
15. Pourmirza, S., Dijkman, R.M., Grefen, P.: Switching parties in a collaboration at run-time. In: 18th IEEE International Enterprise Distributed Object Computing Conference, EDOC 2014, Ulm, Germany, September 1-5, 2014, pp. 136–141 (2014)
16. Pourmirza, S., Peters, S., Dijkman, R., Grefen, P.: A systematic literature review on the architecture of business process management systems. Information Systems **66**, 43 – 58 (2017)
17. Reichert, M., Weber, B.: Enabling Flexibility in Process-Aware Information Systems - Challenges, Methods, Technologies. Springer (2012)
18. Russell, N., van der Aalst, W.M.P., ter Hofstede, A.H.M., Edmond, D.: Workflow resource patterns: Identification, representation and tool support. In: Advanced Information Systems Engineering, 17th International Conference, CAiSE 2005, Porto, Portugal, June 13-17, 2005, Proceedings, pp. 216–232 (2005)
19. Walport, M.: Distributed ledger technology: Beyond blockchain. Tech. Rep. 19, UK Government Office for Science (2016)
20. Weber, I., Xu, X., Riveret, R., Governatori, G., Ponomarev, A., Mendling, J.: Untrusted Business Process Monitoring and Execution Using Blockchain. In: BPM 2016, LNCS 9850, pp. 329–347. Springer (2016)
21. Wood, G.: Ethereum: A secure decentralised generalised transaction ledger eip-150 revision (759dccd - 2017-08-07) (2017). Accessed: 2018-04-08