A Blockchain-Based Compliance System for Business Processes

Lisa Ihde

Business Process Technology Group
Hasso-Plattner-Institute, University of Potsdam, Germany
lisa.ihde@student.hpi.uni-potsdam.de

Abstract. Compliance checking of business processes has various stakeholders. Therefore requirements of internal and external parties have to be checked. This situation is based on trust and can therefore easily be doubted. This paper presents an approach to counteract manipulation based on real-time review of executions on the blockchain. The business process management system sends the current state to the compliance monitor, which responds with triggered and above all violated rules. This approach was implemented and evaluated by a prototype.

Keywords: Business Process Compliance · Process Monitoring · Blockchain · Runtime Verification.

1 Introduction

Business processes are a central component of a company in order to be successful and competitive. One of the most important challenges for today companies is business process compliance. These exist in order to fulfill organizational policies within a company or to comply to legal regulations. For example, there have been many cases of corruption in the financial sector wherefore the Sarbanes-Oxley Act of 2002 [1] was designed to restore the suitability and trustworthiness of business processes. More up-to-date, in terms of data management, a law came out on 25 May 2018. Every company doing business with subjects of the European Union must comply with the General Data Protection Regulations (GDPR) [12]. A company must delete the personal data if they do not have the permission to hold. Thus, there is basically a problem of trust in the implementation of rules, as these may not have been realized correctly or not at all. The paper shows how to combine business process and compliance rules on the blockchain, because blockchain is a trusted way to exchange transaction data over a network of untrusted participants. This means that business processes can be executed on the blockchain as smart contracts [2] without trusting a central authority or a specific participant.

The paper continues with a brief introduction to BPMN, BPMN-Q, business process compliance and blockchain in section 2. Section 3 contains the details of the approach presented in this paper. Related work is considered in section 4 and section 5 evaluates the approach using a working prototype.

2 Preliminaries

For the implementation of business process compliance on the blockchain an example of a business process is needed. This is modelled with the help of BPMN. In addition, rules are formulated for compliance and these are expressed with the help of BPMN-Q.

2.1 Running Business Process Example

The Business Process Model and Notation (BPMN) is a graphical specification language [6]. It provides symbols to model business processes. The notation consists mainly of rectangles, arcs and rhombus. Rectangles represent tasks, arcs represent sequential dependencies between tasks and rhombus represent decisions and mergers. This makes it easy to map complex business processes. For example, the process from the receipt of an order, through payment, to the delivery of the goods (see Fig. 1) for an online-shop.

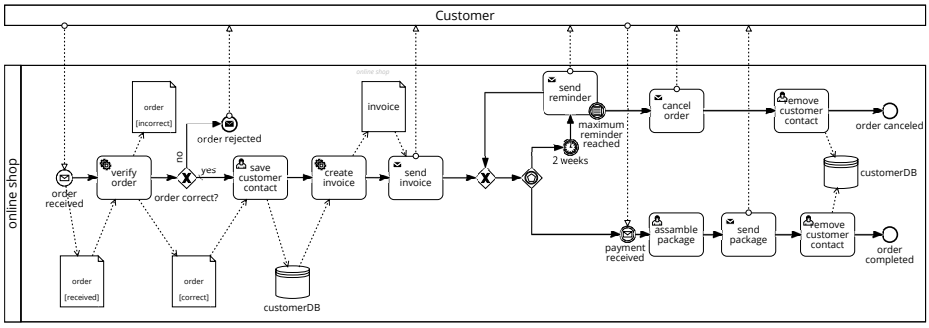


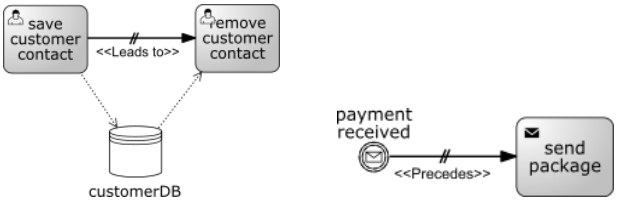
Fig. 1: Order process of an online-shop.

2.2 Business Process Compliance

To achieve compliance with regulations regarding quality management, finances and safety, compliance rules are defined. These rules may be internal policies or describe measures required by law. Thus, these rules help to ensure that a business process is not flawed and meets the requirements of other parties.

2.3 BPMN-Q

BPMN-Q is a visual language that is based on BPMN and it is used to query a business process. [5] We describe a query by connecting two nodes that stand for activities and a path between them.



(a) Sample universality query.

(b) Sample precedence query.

Fig. 2: BPMN-Q patterns.

A business process for an online-shop is given by Fig. 1. According to the GDPR, personal data must be deleted if it is no longer required. Fig. 2a shows a simple query where the task *save customer contact* always leads to *remove customer contact*. This is shown by the expression «Leads to» on the path. There are two types of patterns to distinguish: occurence and order [3]. The first pattern includes three variants:

(1) Universality For this pattern a task always occurs throughout a scope.
(2) Absence This pattern writes to the arrow with an *Exclude(X)* an activity X that is never executed within a scope.
(3) Existence Nothing is written to the arrow and it means that each activity mentioned has the possibility to be executed. But one of the tasks will occur at least once within a scope.

And the second pattern includes two variants:

(4) Precedence This pattern describes a precondition, that must have been executed beforehand within a scope, for a particular activity.
(5) Response It means that one activity will always be followed by the other within a scope.

2.4 Blockchain

Blockchain is a decentralized database or digital cash book [9]. This cash book is copied on thousands of computers in the network. If someone carries out a transaction, it must be confirmed by the majority of the computers in the network and therefore the transaction can also been seen on the other cash books. A chain of data, the blockchain, records each transaction unchanged. If someone wants to change something, this is visible everywhere and the blockchain is therefore a great trust instance. At the same time, as trade-off, all transactions are visible and there is access to this data in order to confirm them.

3 Related Work

There are approaches to enable business process monitoring and runtime verification for choreographies, which could be a basis for compliance checking [10].

A working open-source prototype running on top of the Ethereum blockchain is Caterpillar [7,11].
The core of Caterpillar consists of multiple modules for compiling, modeling, execution, event monitoring and process models. The compilation module generates a smart contract out of a BPMN model, therefore the XML format of the model is needed. The modeler is based on the Camunda BPMN modeler. So all in all Caterpillar has a very extensive and complex code base. Installing a compliance monitor there would cost a lot of time and effort. No dynamic execution of business processes is required for the approach in this paper, because the focus is on the verification of the process through compliance rules.
For the runtime verification for choreographies a smart contract is attached to a process instance. For choreographies, records of the decentralized execution of a process are the basis for the verification. However, this approach uses the first generation of blockchain, the Bitcoin blockchain, where implementation through smart contracts would not yet be possible.

4 Blockchain-based Compliance Checking

This section discusses the research problem addressed in the paper and the background of blockchain technology as a solution.

4.1 Challenges Of Blockchain-based Business Process

Blockchain is the technical basis for crypto currencies, but can also contribute to the improvement or simplification of transaction security in distributed systems compared to central systems [9]. One of the first applications of blockchain is the crypto currency *Bitcoin*. The cryptographic procedure ensures that the blockchain cannot be changed subsequently. This guarantees protection against manipulation. The revolutionary thing about the blockchain is the possibility that now real, direct transactions are possible between two actors who do not know and trust each other without central administration.

With Caterpillar[1] it is possible to execute a business process on the Ethereum blockchain performed by smart contracts, which are generated by a BPMN-to-Solidity compiler [7]. Caterpillar also provides a REST API to request the state of a process and a Modeling Tool, developed on top of Camunda's BPMN modeler[2], to model and execute a process.

4.2 Challenges Of Business Process Compliance

In order to ensure the compliance of a business process, the rules are available as BPMN-Q queries. Both pattern, order and occurence, are basically binary dependencies, where the first node is the precondition or trigger for the second node.

[1] https://github.com/orlenyslp/Caterpillar
[2] https://camunda.com/download/modeler/

Therefore, you have to query the execution traces of running process instances in order to immediately detect a violation or alternatively check it afterwards, when the process achieve a final state.

Limitations The checking of the rules therefore depends on the states and therefore on the activities executed in the process. We differentiate between two possible points in time to check [8]:

- Design Time: The idea is to verify the compliance in the early stages of a process design. The aim is to check every possible rule. It is more a model checking and too computational heavy to simulate every possible execution on the blockchain.
- Runtime: This is a continuous monitoring of the running process instances in order to detect deviating behavior. This allows the triggering activities to be recorded and the corresponding compliance rules to be activated until an activity is executed that deactivates these rules again.

4.3 Overview Of The Approach

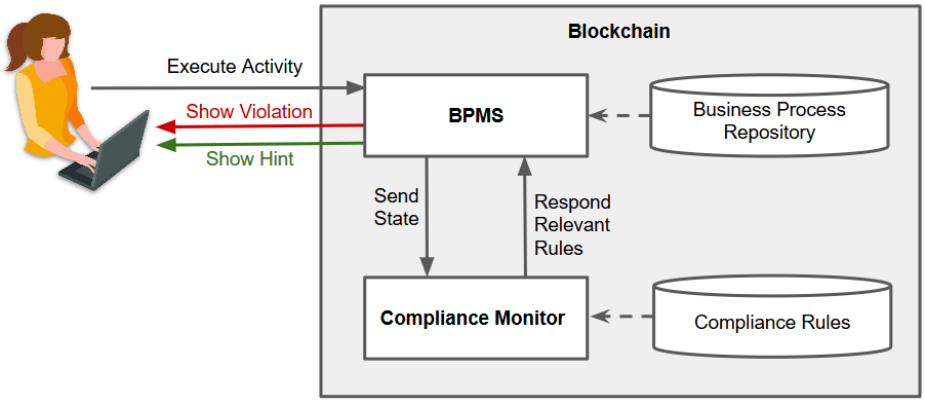


Fig. 3: Approach.

The various compliance rules queries numerous of demands on the business process. To check all these rules would require that all instances of the business process are executed, which corresponds to the design time and is not realizable. A feasible approach is to check during runtime (see Fig. 3). Thus a continuous check reveals a violation of the rules. But in addition to this goal, the vision of the approach is to control behavior and prevent rules from being violated. For this the approach is that the BPMN-Q query is displayed visually when the rule has been activated. By displaying activated rules, it quickly becomes clear which future decisions have to be made and which activities have to be executed. The

BPMN-Q query disappears again if it has been deactivated.

The business process should be executed on the Ethereum blockchain. The Business Process Management System (BPMS) executes the activity of an instance of a business process. The resulting state is sent to the Compliance Monitor. This informs all compliance rules about the current state. If a rule has been activated, this information is sent from the Compliance Monitor to the BPMS and the corresponding activated rule is displayed as hint. This process is repeated with the next executed activity. As a result, the Compliance Monitor can send further activated rules to the BPMS or even violated rules if a final state has been reached and activated rules still exist. If the latter happens, a message about the rule violation is displayed in addition to the hint.

Thus, during the execution a possible violation could be present and pointed out. This deliberately helps to adhere compliance rules. By executing on the blockchain each party has access to the executions and can check whether the corresponding requirements have been implemented and adhered to. This approach solves the trust-based problem between parties and helps to prevent rule violation.

5 Evaluation

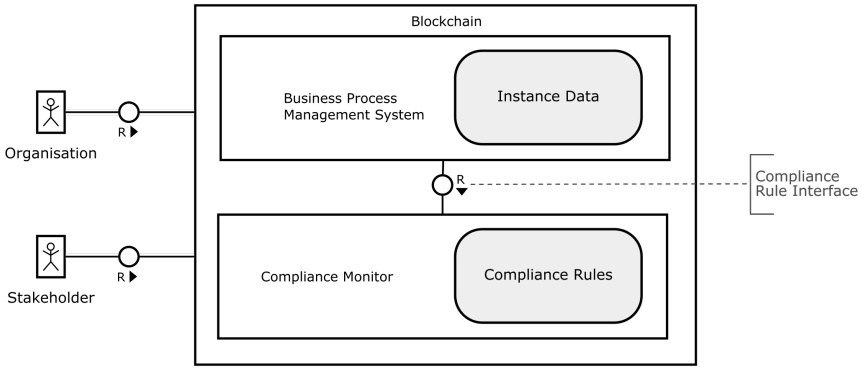


Fig. 4: Architecture as FMC.

A prototype was implemented to evaluate the feasibility of the approach. The technology used is the truffle suite[3], which contains three components: *truffle*,

[3] https://truffleframework.com/

ganache and *drizzle*. They use the *Ethereum Virtual Machine (EVM)* for development and testing. In addition, there are possibilities to deploy contracts on the blockchain and a front-end library to easily create a web app interface. In the following we will first discuss the architecture, then the features and finally a discussion about limitations.

The implementation consists of three major components. The first component is the Business Process Management System. The system is implemented as a smart contract and deployed on the solidity blockchain. The contract is modeled to represent the behavior given by Fig. 4. The contract provides functions to handle inputs that result in state changes, if valid. Furthermore it implements means to externally read the state of the system.

The file-ending sol of the contract indicates, that the used programming language is solidity. Solidity is the common language used for smart contracts on the Ethereum blockchain. The contract defines an enum to store the current state of the System. Furthermore it stores a numeric value. This numeric value is set when an order is received. The value is used later to check, if the values is high enough to proceed with the order, thus determine the following state. To handle the different inputs, every state change is implemented in one function. These functions are marked as external. This means, that these functions can be called by an external instance. Furthermore these functions have the modifier *isInState*. This modifier checks, if the current state is the right state to proceed to the next requested state. If not the an error is returned. Most of the functions follow the same simple pattern: First check that the required state is the current state and then switch to the new state. Some other functions have special checks to determine in what state to switch depending on the current state or user input. For example the *verifyOrder* function first checks that the current state is the initial state. After the check passes, it checks the integer payment for being greater equal than 30. If it is below that value the next state is ORDER_REJECTED. That states represents that the order has failed and is aborted. If not the next state is ORDER_VERIFIED. That state represents that the order is checked and confirmed to proceed. Another interesting function is the *twoWeeksPassedEvent* function. This function checks if the current state is INVOICE_SEND. That Staes indicates that the invoice has been send out to the customer.If that is the current state, it sets the new state to TWO_WEEKS_PAST. This state means that two weeks have passed without receiving a payment from the customer and will lead to sending a reminder to the customer. If the current state is not INVOICE_SEND the function will check if the current state is REMINDER_SEND. That state means that the reminder has been send to the customer. If after that two weeks pass, the order will be ultimately terminated. That is initialized by the state TWO_WEEKS_PAST_. Combining all the functions, states and modifier this rather simple smart contract is able to model this simple Business Process.

The second component is the compliance monitor. To provide trust this component has to be implemented as a smart contract, containing one interface class,

on the blockchain. To implement different rules, concrete implementation of the interface class has to be implemented. The interface class is named *ComplianceRule*. This class contains two functions that needs to be implemented for each rule. The first function is a *reset* function, that is used to reset the checker if it runs into a state that cannot be left or some other unpredicted case appears. The other function is called *checkComplianceRule*. This function takes as input an integer with the current state as enum. The function then emits events that are containing two boolean. The first boolean is *active*. That boolean is true, when the rule is considered active. Active means that the user should be indicated with additional information to mind the rule while proceeding with the interaction of the system. The second boolean is called violation. This boolean states if the compliance rule was violated. If violation is true, an violation message should be stated to the user and the system should inform the user that a major mistake have been made. A concrete implementation can be seen for the precedence example compliance rule (see Fig. 2b). The *reset* function is simply setting the boolean active and violation to false. The *checkComplianceRule* function is called every time a state change is happening with the new state as input. The function checks, if the new state is the enum for INVOICE_SEND(7). If this is true the boolean *active* is set to true. This state is the trigger for the compliance rule to be relevant and been shown to the use. If the system reaches the states 8 or 19, the enums for PAYMENT_RECEIVED and TWO_WEEKS_PAST_, the boolean *active* is set back to false. In case the state PACKAGE_SEND is reached while active, *violation* will be set to true, because the compliance rule was violated. At the end of the function, both boolean will be emitted together with the received enum for the new state. The new state is also emitted mainly for debugging reasons, but can also be used to synchronize the events with the state changes.

The third component is the user-interface. The user-interface provides the user with visual means to interact with the BPMS. Furthermore it provides the user with visual output about the state of the different compliance rules. Thus the interface is able to pass user interactions to the BPMS, display the current state of the system. Additionally it sends the state of the system to the compliance monitor and receives the emitted events to display hints and violations if needed to the user. The interface was implemented by using drizzle in combination with react. The important file is the App.js file. The chose programming language, as hinted by the file ending, is javascript. Because the states are passed as numbers between the user interface and the blockchain, the frontend needs to implements the enum again. Being able to translate the enums helps to communicate with the svg and the user. The user is presented mainly with the stated svg. This svg contains the business process in a graphical representation, while using the different nodes as buttons. Furthermore the user has an input field and a reset button. The input is used for getting the initial value of the order. But before this is shown to the user, the component first waits for drizzle to be loaded and running, to communicate with the blockchain and also loads the different implementations of the compliance rules and initializes

the event communication, as implemented in the *componentDidMount* function. If the user then clicks on one of the highlighted buttons, the component will invoke the specific function on the BPMS. After waiting for the new state to be set, the frontend will log the new state, pass it to the compliance monitor and adjust the frontend and display hints or violations if needed. For the logging, the function *logState* is being called at the of every state changing function, for example *activateVerifyOrder*. The function reads the new state, saves it inside the *orderOfState* array and calls the *updateChecker* function. This function calls for every known implementation of the a compliance rule the *checkComplianceRule* function with the new state. The events will that are returned will be handled as implemented in the *startComplianceChecker* function. The function will be called once after drizzle is loaded. Taking the universality example (see Fig. 2a), it is seen, that on an incoming event, the values for violation and active are being checked. If the vioation value is true, an violation message will be shown, if the active value is true, the compliance rule as a picture will be shown in the interface. Additionally the *render* function is called repeatedly. This function uses the help function *returnState* to get the current state of the system. Using the current state the function repeatedly updates the interface and disables and enables buttons, inside the svg.

Finally, a screencast video is available[4].

6 Conclusion and Discussion

We can execute a business process on the blockchain, but the corresponding smart contracts must be created manually. We use the svg export of the BPMN to use it as interactive business process. This is done by connecting the IDs of activities with the states of the smart contract. So now we have clickable activities. In order to use compliance rules, there is an interface class and at least one rule for each pattern of BPMN-Q query. This makes it very easy to add further rules as smart contracts. In addition, hints are displayed in the form of BPMN-Q and rule violations are displayed as text messages. The biggest limitation is that the creation of smart contracts for the business process and rules is not automated. There are projects such as Caterpillar [7], which enables automation of the business process into a smart contract. But this is a research prototype, which of course has some limitations and dependencies. For the automation of compliance rules the LTL notation [4] to generate smart contracts can be used. The main benefit of this approach is that the blockchain is tamper-proof and there is a conscious behavior by executing activities. Thus violations of rules can be prevented. But on the other hand, the transactions are costly and the verification of internal policies could be leave out.

[4] https://youtu.be/RIsA2bxh0TU

References

1. Sarbanes-Oxley Act of 2002. Public Law 107–204, (116 Statute 745), United States Senate and House of Representatives in Congress, 2002.
2. Weber I., Xu X., Riveret R., Governatori G., Ponomarev A., Mendling J. (2016) Untrusted Business Process Monitoring and Execution Using Blockchain. In: La Rosa M., Loos P., Pastor O. (eds) Business Process Management. BPM 2016. Lecture Notes in Computer Science, vol 9850. Springer, Cham
3. Liang Song, Jianmin Wang, Lijie Wen, and Hui Kong, Efficient Semantics-Based Compliance Checking Using LTL Formulae and Unfolding, Journal of Applied Mathematics, vol. 2013, Article ID 962765, 24 pages, 2013. https://doi.org/10.1155/2013/962765.
4. Awad A., Decker G., Weske M. (2008) Efficient Compliance Checking Using BPMN-Q and Temporal Logic. In: Dumas M., Reichert M., Shan MC. (eds) Business Process Management. BPM 2008. Lecture Notes in Computer Science, vol 5240. Springer, Berlin, Heidelberg
5. A. Awad, BPMN-Q: a language to query business processes, in Proceedings of the 2nd International Workshop on Enterprise Modelling and Information Systems Architectures (EMISA 07), M. Reichert, S. Strecker, and K. Turowski, Eds., pp. 115–128, St.Goar, Germany, October 2007.
6. OMG, Business Process Model and Notation (BPMN) ver. 2.0, January 2011, http://www.omg.org/spec/BPMN/2.0.
7. López-Pintado, Orlenys et al. Caterpillar: A Blockchain-Based Business Process Management System. BPM (2017).
8. Hashmi, M., Governatori, G., Lam, HP. et al. Knowl Inf Syst (2018) 57: 79. https://doi.org/10.1007/s10115-017-1142-1
9. Puthal, Deepak & Malik, Nisha & Mohanty, Saraju & Kougianos, Elias & Das, Gautam. (2018). Everything You Wanted to Know About the Blockchain: Its Promise, Components, Processes, and Problems. IEEE Consumer Electronics Magazine. 7. 6-14. 10.1109/MCE.2018.2816299.
10. Mendling, Jan & Weber, Ingo & Aalst, Wil M. P. & Brocke, Jan vom & Cabanillas, Cristina & Daniel, Florian & Debois, Søren & Di Ciccio, Claudio & Dumas, Marlon & Dustdar, Schahram & Gal, Avigdor & García-Bañuelos, Luciano & Governatori, Guido & Hull, Richard & La Rosa, Marcello & Leopold, Henrik & Leymann, Frank & Recker, Jan & Reichert, Manfred & Zhu, Liming. (2018). Blockchains for Business Process Management - Challenges and Opportunities. ACM Transactions on Management Information Systems. In press, accepted. 10.1145/3183367.
11. Christoph Prybila, Stefan Schulte, Christoph Hochreiner, and Ingo Weber. 2017. Runtime Verification for Business Processes Utilizing the Bitcoin Blockchain. arXiv report 1706.04404. arXiv. https://arxiv.org/abs/1706.04404
12. Ian Gotts, 14.02.2019, http://www.bpminstitute.org/resources/articles/gdpr-process-issue