Certification of Business Processes and Workflows via Blockchain

Alberto Leporati

University of Milan-Bicocca, Department of Informatics, Systems and Communication, Edificio U14 (ABACUS), Viale Sarca 336, 20126 Milano, Italy

Abstract

Inspired by the functioning of software applications that allow administrative staff to manage different types of bureaucratic processes, we propose to use permissioned blockchains to implement administrative and business processes involving different organizations. We describe the design of our system, comprising its architecture, the functional analysis, and a discussion on how to implement the required features. Such a design is the first step towards the implementation of a blockchain-based system for the certification of business processes and workflows.

Keywords

Blockchain, Business Process Modeling, Certification of Business Processes, Certification of Workflows

1. Introduction

A blockchain is a distributed ledger shared among a computer network's nodes. The nodes work together to maintain a secure and decentralized record of information, stored inside blocks. Each block is linked to the previous one through a cryptographic hash function. This makes the contents of a block virtually impossible to be changed, since a modification would make the cryptographic check of hash fingerprints to fail.

Typical uses of blockchains include supply chain management [1], document notarization [2], decentralized finance (DeFi) applications [3], and tokenization of digital and physical assets [4]. Every application needs a blockchain with certain characteristics, and in fact there are different types of blockchains. First of all, a blockchain can be *public* or *private*, depending upon whether its contents is publicly available or not. Furthermore, a blockchain can be either *permissionless* or *permissioned*. In the former case, anyone can join the network that maintains the blockchain, holding an entire copy of it. Usually, this also means that anyone can participate in validating the information contained into the blocks. To prevent malicious behaviors, permissionless blockchains usually adopt a majority-based consensus algorithm, which may be however very inefficient in terms of computational power and energy consumption. In the case of permissioned blockchains, instead, only authorized actors can be part of the network that manages the blockchain. Permissions are established based on the possession of cryptographic credentials, managed by a certification authority. In fact, permissioned blockchains are generally

6th Distributed Ledger Technology Workshop (DLT 2024), May 14-15, 2024, Turin, Italy

alberto.leporati@unimib.it (A. Leporati)
0000-0002-8105-4371 (A. Leporati)

managed by *consortia*, made up of entities that potentially do not trust each other, but who must collaborate to achieve a common goal. Permissioned blockchains do not require a real consensus algorithm, as they are used as distributed digital ledgers. The members of the consortium simply write their declarations in the register, and these declarations can no longer be deleted or modified. The other members of the consortium simply observe the statements made, and assume they are true; this creates trust between the parties. Subsequently, if some of the statements made turn out to be false, the members who made them will be punished by the consortium, for example with expulsion. A problem of permissioned blockchains is that they are completely controlled by the consortium. This means that if the consortium members agree, they can modify the content of the blocks as they want. This problem is especially felt for small consortia. An often adopted solution to prevent this possibility is to save the cryptographic hash of the last block of the blockchain on a well-known public blockchain, such as Polygon[1], at established time intervals.

Another common use of blockchains is the *notarization* of documents. The idea is to demonstrate that a given digital document existed in exactly that form at a given time. Since the block size does not allow documents to be saved directly in the blockchain, the document file is usually saved on a decentralized file system, such as IPFS[2], while the document's cryptographic fingerprint — calculated via a hash function — is saved on the blockchain. When the documents on which notarization is performed are certifications — for example certificates of origin of raw materials, or quality certificates of processed products — we sometimes speak of *document certification* rather than notarization. A natural next step is the certification of processes, or workflows. Inspired by the functioning of software applications that allow administrative staff to manage different types of bureaucratic processes, such as for example Archiflow[3] or Cineca's Titulus, in this paper we propose to use permissioned blockchains to implement administrative and business processes involving different organizations. An example of such processes could be a bureaucratic process that involves the collection of documents, signatures and authorizations from various bodies. Or, in the industrial sector, the certification of the processing phases of a product, collecting information both from the machinery and from the people involved in controlling the process.

2. Some Related Works

Of course, we are not the first to propose modeling business processes or workflows via blockchains. A comprehensive literature review is contained in [5]. There, blockchains are identified as an emerging technology in the field of Business Process Management (BPM); it is claimed that, through smart contracts, a blockchain may enforce the obligations of counterparties that transact in a business process. The paper focusses on how trust is maintained during the entire lifecycle of a business process, and identifies some challenges and research directions. It is also pointed out that most of the reported applications are at their primary stages, thus further research efforts are needed to meet the technical challenges posed by them.

[1]https://polygon.technology/
[2]https://ipfs.tech/
[3]https://www.siav.com/archiflow/

In [6], it is shown that a process model comprising tasks performed by multiple untrusted parties can be coordinated via smart contracts operating on the blockchain, without requiring a central authority. However, the cost required for blockchain use is highly dependent on the volume of data recorded and the frequency of data updates by smart contracts. The authors then propose an optimized method for executing business processes on top of commodity blockchain technology. The method is empirically compared to a previously proposed baseline, by measuring resource consumption and throughput.

In [7], it is reported that 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. The authors then present the conceptual foundations of model-driven approaches for blockchain-based collaborative process execution, and they compare two concrete approaches, Caterpillar [8] and Lorikeet [9]. Both tools are proof-of-concepts, proposed to work on the Ethereum public blockchain. The effectiveness of the former is demonstrated through an industrial use case.

As we can see, many projects proposed in the literature are at an experimental level. They are mostly based on the Ethereum public blockchain, which is too expensive to be used in real-world applications. Furthermore, a public blockchain makes all processes visible to anyone, possibly leaking some relevant information about their operation. Moreover, most of the proposed solutions require that users learn the Business Process Model and Notation (BPMN) to define their processes, which can be hard to learn. The solution proposed in this paper works instead on a private and permissioned version of the Ethereum blockchain, whose state is regularly notarized on a public permissionless blockchain. Moreover, the users will be able to use a very simple graphical interface to define their own processes.

3. Design of the System

The aim of the software is to allow users to define their own processes or workflows, and to certify their execution on a blockchain. Users of the software will typically be companies, who register by creating their own account. The account associated with the company contains the company's data, for billing, communicating with it, etc. Furthermore, this account is the administrator for the accounts of that company subsequently created, and is also the owner of all smart contracts created by the company. The blockchain used will be a permissioned (and possibly private) version of Ethereum, such as Quorum[4]. The blockchain will be used as a distributed ledger, using proof-of-authority (PoA) as a consensus algorithm.

The company administrator can create *roles*, *users*, and assign their users to the created roles. Each role is associated with a set of possible *operations*, or *actions*; when the administrator creates a role, it chooses from a list which of these operations are allowed for that role. Examples of roles could be: head of legal department, quality manager, production manager, etc. Examples of operations that can be performed are: uploading a document, signing a document, approving a process status. In the first version of the software, the list of allowed operations will be predefined; the software will allow a company administrator to create and name any role,

[4]https://github.com/ConsenSys/quorum

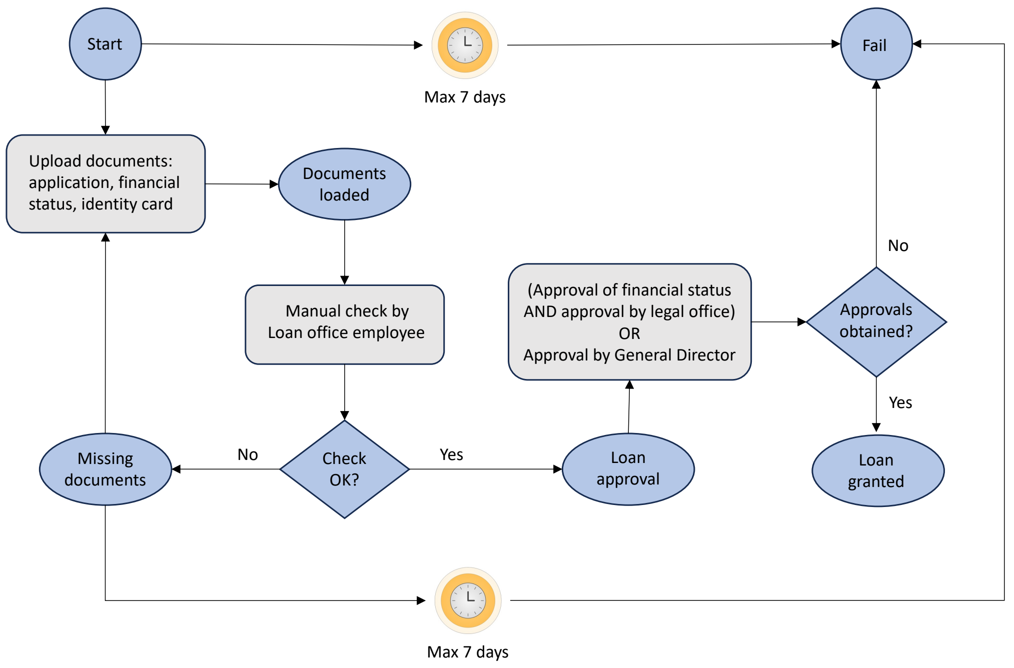
assigning it a set of allowed operations taken from that list. The administrator can then create one or more users, assigning them this role.

To define a process, the user uses a graphical interface, that allows one to arrange graphic elements on a canvas, which represent states, conditions and transitions between the states of the process. The process is represented through a directed graph, whose nodes are the *states* in which the process can be found; at any instant, the process can be in one and only one state. Each process or workflow starts from a single *initial state*. On the other hand there may be one or more *final states*, in which the process can terminate successfully. And for every process there is always a state indicating that the process terminated *unsuccessfully*, for example because it was invalidated by its owner.

The directed arcs correspond to *state transitions*. A state transition can occur if a predefined *list of conditions* is satisfied by an appropriate set of *actions*. These actions can occur in any order; in fact, if there is a pre-established order in which the actions must take place, it will be necessary to modify the process in such a way that a change of state occurs in correspondence with each action, enabling the subsequent ones. Examples of conditions that must be satisfied to move to a next state include: (1) uploading one or more documents; (2) digitally sign one or more documents; (3) the approval of the current state by one or more people, or by one or more roles. In the former case, approval is required from the specified people, while in the latter case it is sufficient for anyone in that role (for example, any employee of the legal department) to give their approval; (4) a predefined amount of time has passed, within which the prescribed actions had to be carried out. In this case, having verified that only some of these actions have been performed, the process can go into a state in which alternative measures are taken (or, in the worst case, into the state indicating that the process has failed).

An example of the business processes/workflows we want to consider is illustrated in Figure 1. The example concerns the request for a loan by a user. To avoid overly complicating the diagram, we have left out some details, such as the roles involved and details on the precise actions to be carried out (for example, the checks performed by the loan office employee). This information will be visible in our application, by selecting the nodes or arcs of the diagram. In Figure 1, states are represented by circles or ovals, choices are represented by diamonds, and the conditions for moving from one state to another are indicated inside rounded rectangles. Time conditions are indicated with a clock icon. In particular, the workflow operates as follows:

1. The process starts from an initial state, labeled “Start” in the automaton of Figure 1.
2. A user, who is requesting a loan, must upload the application and a document certifying their financial status. This application can be signed digitally, or it can be signed by hand; in this case, a copy of the identity card must be attached. When these actions are completed, the process goes into the “documents loaded” state.
3. A manual check is made by a loan office employee to ensure that the requested documentation has been provided correctly, with digital signature, or hand signature. If the check is successful, the employee provides authorization to proceed and the process moves to the “loan approval” status; if the check fails, the process switches to the “missing documents” state, and a notification is sent to the user. When the user uploads the missing documents, the process will return to the “documents loaded” state, and the employee can perform the check again. If the user does not upload the missing documents within 7 days, or if



Start
Max 7 days
Fail
Upload documents: application, financial status, identity card
Documents loaded
Manual check by Loan office employee
(Approval of financial status AND approval by legal office) OR Approval by General Director
Approvals obtained?
No
Yes
Missing documents
No
Check OK?
Yes
Loan approval
Loan granted
Max 7 days

Figure 1: An example of workflow, for a user requesting a loan

they never uploaded any document, the process ends unsuccessfully.

4. To approve the loan, the following two conditions must be met: (1) the office that performs the checks on the user's financial status must provide its authorization, and (2) the legal department must provide its authorization. Alternatively, the authorization of the General Director is sufficient. If any of these permissions are denied, the process ends unsuccessfully and the user is notified. Otherwise, the process ends in a successful final state, and the loan is granted.

The roles necessary to carry out this process (not shown in Figure 1) are the following: (1) Normal User (the users who request the services); (2) Loan Department (all employees who carry out the initial checks on documents uploaded by users belong to this role); (3) Financial Department (all office employees who carry out checks on the users' financial status belong to this role); (4) Legal Department (all office employees who carry out checks on the validity of documents from a legal point of view belong to this role); (5) General Director: only one person belongs to this role, namely the General Director.

Let us note that authorizations are associated with roles rather than directly with people. However, the system will record the name of the people who provided the authorizations, for liability reasons. Authorization conditions can in theory be very complex; for example, in a series of authorizations, it may be required that at least two employees of an office sign or give their authorization, or that these two authorizations be replaced by that of their office manager.

4. (Future) Implementation

The developed software, currently, in the design phase, will be a web application. As such, it will allow users of the service to register and authenticate. It will also allow the administrator of each company or institution to create their own roles, users and processes. The software architecture of the proposed system consists of the following elements:

- The web application user interface (frontend), that communicates with the backend via REST APIs. It will be implemented using the React library[5].
- A backend. This is the heart of the software: it contains the operating logic, manages the information stored in the centralized database, exposes the REST APIs to the frontend and to the outside, and manages access to the blockchain and its smart contracts. It will be implemented using the Sails Javascript MVC framework[6]. Communication between the APIs and the blockchain will occur through the Web3.js Ethereum Javascript library[7].
- A centralized database, in which all information that should not be recorded on the blockchain is stored. This includes information contained in company accounts, billing information, etc. To this aim, MongoDB[8] will be used.
- A permissioned (and possibly private) version of the Ethereum blockchain, such as Quorum. Since the documents cannot be saved in the blockchain, due to the limited size of the blocks, they will be saved on a private IPFS network instance, while their hash fingerprints will be saved on the blockchain.
- A public blockchain (such as Polygon), on which the hash fingerprint of the last current block of the permissioned blockchain will be notarized at regular time intervals.

Access to the blockchain always goes through the REST APIs. These control the permissions to create or invoke smart contracts, as well as the permission to write information on the blockchain. The presence of REST APIs will also allow external developers to integrate other software applications with the system proposed here, according to the principle of process/workflow Certification-as-a-Service.

Each process/workflow defined by an organization will be transformed into a smart contract written in the Solidity language by a specific Process-to-Solidity compiler. Basically, a process/workflow defines a deterministic finite state automaton (DFA) that will be simulated by the smart contract. The history of all the states the DFA has been in, starting from the initial state, is stored in a Solidity mapping. The smart contract also knows the structure of the DFA, and therefore not only the state diagram but also all the conditions associated with the transition from one state to the next. Note that this information can be stored and managed in (at least) two ways. A first way, which we could call *uniform* by analogy with what happens with some theoretical computation models [10], provides that all the information relating to the DFA that encodes the process is stored in a smart contract in appropriate data structures (mapping, arrays, etc.). The smart contract is therefore unique, and can simulate any process. By analogy with what happens with Turing machines, we could say that it is a *universal* smart

[5]https://react.dev/
[6]https://sailsjs.com/
[7]https://web3js.org/
[8]https://www.mongodb.com/

contract. The second way, which we could instead call *semi-uniform*, requires that the structure of the smart contract itself (i.e., its control structures: loops, branches, etc.) reflects the structure of the process/workflow (i.e., the DFA) to be simulated. Both choices have pros and cons. In the former case, there is a single smart contract to deal with, which can be optimized by hand; the Process-to-Solidity compiler will "only" have to take care of generating the correct data structures to represent the given process (DFA) within the smart contract. Naturally, the smart contract will be quite complicated, having to be able to simulate every possible case described within the data structures. On the other hand, in the latter case the smart contracts that encode the processes (DFA) will certainly be simpler, but the Process-to-Solidity compiler will be more complicated. We are currently oriented towards this last option.

In any case, the smart contract must keep track of all the actions that have been performed to verify the conditions. Thus, in the example reported above, the smart contract can — at some point in the execution of the process — store the fact that the current state is "loan approval", which means that it expects an authorization from part of the Financial Department, and an authorization from the Legal Department (or, alternatively, by the General Director). Assuming that only the authorization from the Legal Department has been received, the condition for the transition to the "loan granted" status will only be partially verified. At this point the second authorization could arrive, or the authorization from the General Director could arrive, which makes the first authorization received useless (but does not delete it from the state of the smart contract). If one of these possibilities occurs, the smart contract notices that all conditions are verified, and records the change in state of the automaton. The communication of an action to the smart contract, the query about which conditions remain to be verified, and the query about the current state of the automaton, all occur through appropriate calls to functions written in Solidity. At any time, a call to the `abort` function can bring the process to an unsuccessful final state, effectively causing the execution of the process to fail. Clearly, the `abort` function can only be invoked by those authorized to do so, typically the owner of the smart contract. Similarly, if we realize that the process/workflow is blocked because one of the conditions required in the current state can never be verified, it is possible to return the automaton to one of the previously assumed states. Clearly this means that some authorizations acquired and some actions performed will be discarded (but will remain in the history of the states of the smart contract), given that the smart contract will have to be in a state consistent with the current state of the automaton. Note also that the process can only return to one of the previously assumed states, and not to an arbitrary state.

5. Conclusions and Directions for Future Work

Inspired by some software applications for document flow management, in this paper we proposed a blockchain-based system for the certification of processes and workflows. The system is currently in the design phase, and in this paper we have provided some of the ideas on which it is based. In addition to application development, we will evaluate whether to adopt the Business Process Model and Notation (BPMN)[9] as a format for representing process procedures. This would allow for more direct interfacing with softwares that already adopt this standard.

[9]https://www.bpmn.org/

Another future challenge will be to ensure that different organizations can independently provide their own processes, which communicate with the processes of other organizations, all of them being certified by our system.

A more immediate and practical problem concerns the maximum size and complication of the processes that can be represented by smart contracts. In fact, in November 2016 EIP-170[10] imposed a smart contract size limit of 24.576 bytes, to prevent denial-of-service (DOS) attacks. One immediate solution would be to divide large smart contracts into smaller ones; however, this requires to design beforehand the ways in which large contracts can be split. A technically more complicated solution would be to use a proxy system[11] with DELEGATECALLs, which execute another contract's function with the state of the calling contract.

References

[1] F. Dietrich, Y. Ge, A. Turgut, L. Louw, D. Palm, Review and analysis of blockchain projects in supply chain management, Procedia Computer Science 180 (2021) 724–733. Proceedings of the 2nd International Conference on Industry 4.0 and Smart Manufacturing (ISM 2020).

[2] M. J. M. Chowdhury, A. Colman, M. A. Kabir, J. Han, P. Sarda, Blockchain as a Notarization Service for Data Sharing with Personal Data Store, in: 17th IEEE Intern. Conf. On Trust, Security And Privacy In Computing And Communications/ 12th IEEE Intern. Conf. On Big Data Science And Engineering (TrustCom/BigDataSE), 2018, pp. 1330–1335.

[3] Y. Chen, C. Bellavitis, Blockchain disruption and decentralized finance: The rise of decentralized business models, Journal of Business Venturing Insights 13 (2020) e00151.

[4] G. Wang, M. Nixon, Sok: tokenization on blockchain, in: Proceedings of the 14th IEEE/ACM International Conference on Utility and Cloud Computing Companion, UCC '21, Association for Computing Machinery, New York, NY, USA, 2022, pp. 1–9.

[5] W. Viriyasitavat, L. D. Xu, D. Niyato, Z. Bi, D. Hoonsopon, Applications of Blockchain in Business Processes: A Comprehensive Review, IEEE Access 10 (2022) 118900–118925.

[6] L. García-Bañuelos, A. Ponomarev, M. Dumas, I. Weber, Optimized execution of business processes on blockchain, in: J. Carmona, G. Engels, A. Kumar (Eds.), Business Process Management, Springer International Publishing, Cham, 2017, pp. 130–146.

[7] C. Di Ciccio, A. Cecconi, M. Dumas, L. García-Bañuelos, O. López-Pintado, Q. Lu, J. Mendling, A. Ponomarev, A. Binh Tran, I. Weber, Blockchain Support for Collaborative Business Processes, Informatik Spektrum 42 (2019) 182–190.

[8] O. López-Pintado, L. García-Bañuelos, M. Dumas, I. Weber, A. Ponomarev, Caterpillar: A business process execution engine on the Ethereum blockchain, Software: Practice and Experience 49 (2019) 1162–1193.

[9] A. Binh Tran, Q. Lu, I. Weber, Lorikeet: A Model-Driven Engineering Tool for Blockchain-Based Business Process Execution and Asset Management, in: International Conference on Business Process Management, 2018, pp. 56–60.

[10] N. Murphy, D. Woods, Uniformity is Weaker than Semi-Uniformity for Some Membrane Systems, Fundamenta Informaticae 134 (2014) 129–152.

[10]https://eips.ethereum.org/EIPS/eip-170

[11]https://docs.openzeppelin.com/upgrades-plugins/1.x/proxies