Blockchain platform for executing Collaborative Business Processes with Hyperledger Fabric

Lucas Pairetti[1], Tomás D'Anunzzio[1], Mercedes Canavesio[1], Pablo Villarreal[1]
[1]CIDISI - Universidad Tecnológica Nacional Facultad Regional Santa Fe, Santa Fe, Argentina
{lpairettirebotta, tdannunzio, mcanavesio, pvillarreal}@frsf.utn.edu.ar

Resumen

Blockchain and Distributed Ledger Technologies have been identified as a tool to build and increase trust in collaborative business processes as well as to provide integrity, security, and transparency to data. Permissioned or private blockchains are more suitable for collaborative processes since the focus is on an efficient exchange of data and transaction execution. In this work, we propose a permissioned blockchain platform for the implementation and execution of collaborative processes, which is based on HyperLedger Fabric (HF). Smart contracts are defined and implemented on the blockchain platform for performing the validation of the messages sent by the organizations as part of the choreography of interactions agreed on collaborative processes. The proposed platform enhances data privacy by using channels for the processes, and the proposed approach for smart contracts allows a blockchain completely decentralized for managing process instances of collaborative processes.

Introduction

Organizations are focusing on establishing cross-organizational collaborative networks with the purpose of improving their performance and competitiveness. The behavior of a cross-organizational collaboration is agreed on the collaborative business processes that define the global view of the interactions among organizations to achieve common business goals [17],[19]. The digitalization (implementation and execution) of collaborative processes requires appropriate Information Technology to fulfill the following requirements:

- Autonomy: Organizations behave as autonomous entities, hiding their internal decisions, activities, and processes.
- Decentralized management: Collaborative processes are jointly managed by the organizations.
- Peer-to-Peer interactions: Organizations interact in a direct way without the mediation of a third party.
- Privacy: Organizations reveal only the information that is shared in the interactions, and this public information is expected to be accessible only to the interested parties of the network.
- Trust: The participants of collaborative networks are known. Integrity, security, and transparency of the shared data are important to achieve an appropriate trust level among the organizations.
- Execution of agreed interactions: the business logic defined in the business solutions must be fulfilled by the implementation of the technological solution.

There are several technologies such as web services [12], software agents [16], and cloud-based microservices [4, 5] that were proposed to build platforms for implementing and executing collaborative processes. However, although most of the above requirements could be fulfilled by these platforms, the aspects of privacy and trust are not completely covered in these approaches.

Blockchain and Distributed Ledger Technologies have been identified as a tool to build and increase trust in collaborative business processes [15, 9], as well as to provide integrity, security, and transparency to data that is shared [15, 20]. A blockchain allows for capturing the history and current state of collaborative business processes, by recording in each block the transitions of the transactions executed between the parties. This distributed registry is immutable and invulnerable, thus guaranteeing confidence that the stored information has not been deliberately or accidentally altered [6, 22]. Information about the state of the instances of each process can be shared and updated locally on each node [18]. The existence of an immutable public ledger enforces a commonly agreed-upon single source of truth [2, 18].

Some proposals have been proposed that aim to implement collaborative processes using blockchain technology [18] [7] [13]. However, these proposals make use of public or non-permissioned blockchain platforms that are oriented to the exchange of cryptocurrencies (such as Ethereum) in an open network in which the participants do not know each other. In these platforms, the consensus mechanism that is used (such as the Proof of Work protocol) requires large computational resources to mine and validate new blocks. Because collaborative networks are closed in the sense that imply integration and

collaboration among a set of known parties, the focus is not on the mining of complex transactions. Instead, the focus is on an efficient exchange of data and transaction execution. Hence, permissioned or private blockchains are more suitable for collaborative processes.

In this work, we propose a permissioned blockchain platform for the implementation and execution of collaborative processes. Smart contracts are defined and implemented on the blockchain platform for performing the validation of the messages sent by the organizations as part of the interactions allowed in collaborative processes. The proposed blockchain platform is built on HyperLedger Fabric (HF) [10], which is a platform and a framework for building permissioned blockchains. Section 2 presents a background on blockchain technology and HyperLedger Fabric (HF). Section 3 describes the proposed blockchain platform along with its architecture and smart contract approach for executing collaborative processes. Section 4 presents a case study about the implementation of a collaborative process with the proposed platform. Section 5 discusses related works. Section 6 presents conclusions and future works.

Background

Blockchain

A blockchain is a type of distributed ledger technology (DLT) where all transactions are digitized and decentralized. It is a database that records all transactions or events that are executed and shared among network participants [11]. Blockchains allow untrusting members to interact verifiably in a peer-to-peer network without the need for a trusted authority. All transactions are visible to all nodes on the network. Anyone can verify the data and trace the history through a computer on the network to ensure the reliability of the information. This is possible because blocks are added in chronological order and contain a cryptographic hash of the previous block. Thus, the data recorded in the blockchain cannot be manipulated, altered, hidden, or falsified. Any record that is added to the blockchain cannot be deleted and all previous transactions become immutable [11]. Any blockchain user can easily trace any previous transaction by accessing any node in the distributed network, as each of the transactions is validated and recorded with a timestamp.

A blockchain may be private or public [21]. Private (or permissioned) blockchains are made up of registered and known participants and only they can come forward to verify and validate transactions. On the network, the transaction processing rate is very high with very few authorized participants. Therefore, less time is required to achieve network consensus and more transactions can be processed in a second. Private blockchains have very strong data privacy, where any changes can be made simply when all nodes agree that data can be changed by consensus.

Public (or non-permissioned) blockchains have a more limited transaction processing rate. Consensus mechanisms like Bitcoin's Proof of Work (PoW) on public blockchains require the entire network to reach consensus on the status of transactions. Public blockchains are based on an append-only data process that leads to immutable data storage. To ensure the integrity of the blockchain, all blocks are linked to the genesis block. Everyone in the network is incentivized to act according to the contract to achieve the best results for the network. Every single transaction on a public blockchain is open for the public to verify. Public blockchains require large-scale operation costs and the transaction times are high.

Hyperledger Fabric

Hyperledger Fabric (HF) is a Hyperledger project of blockchain frameworks and tools established under the sponsorship of the Linux Foundation in early 2016 [1]. Hyperledger Fabric is a modular (permissioned, private) blockchain platform [10]. This platform has been considered one of the most mature blockchain platforms to date and was the first platform that allows running smart contracts in several general-purpose programming languages (e.g., Node.js, Java, and Go) [1]. The specific feature of Hyperledger Fabric different from other platforms is the order execution-validation architecture The transaction flow in Hyperledger Fabric consists of three phases: transaction execution, ordering, and validation. Unlike public blockchains, all nodes in Hyperledger Fabric have an identity, which can be classified into the following three functions [10]:

- Customer functions: submit a transaction proposal and submit the transaction to place the order;
- Peer roles: execute a transaction proposal, validate the transaction, and maintain blockchains;
- Orderer functions (order service nodes): collect customer transactions and determine the general order of all transactions.

In addition to the above nodes, the following design components are also included in Hyperledger Fabric [10]. The Membership Service Provider (MSP) manages user identities and controls access to the blockchain network. MSP uses a certificate authority (CA) to validate and authenticate users.

A smart contract, also called chaincode, defines the life cycle of an asset that is maintained in the world state. The chaincode contains methods to make a change to the asset and to query the current state of the asset. It is a software component used in Hyperledger Fabric that packages one or more smart contracts for installation on a particular channel. When installing a chaincode, an endorse policy must be defined that decides which peers (organizations) have the right to support a transaction in the smart contract for it to be considered valid and committed in the ledger.

A transaction is an action requested by a client application to be executed to change an asset that is in the current state. A transaction can read or write the current

world state and must be validated according to the endorse policy defined in the chaincode.

A ledger is a non-changeable journal of all transactions that occur in a channel. By querying the ledger, an authorized client can view the chain of transactions from the beginning of an asset to the current state of the asset.

World status is the current state of each asset in the ledger. By querying the world state, a client can acquire the current state of an asset.

Different channels can be created in Hyperledger Fabric, which provides a separate communication layer for a subset of participants to maintain communication and data privacy [10].

Blockchain Platform for Collaborative Business Processes

The proposed blockchain platform for implementing and executing business processes is based on a permissioned blockchain. It makes use of smart contracts to validate the transactions that are triggered by applications for the sending of messages from one organization to another, as part of the interaction flow or choreography defined in the collaborative processes. The distributed ledger storages the messages exchanged by the organizations. A message contains the public or shared information that a sender (the organization that triggers the transaction) sent to a recipient (the organization that expects to receive a message). Smart contracts validate that the order of the messages is respected as it was defined in the choreographic logic of the collaborative processes.

Architecture of the Blockchain platform for collaborative processes

The architecture of the proposed platform is defined based on the elements provided by HyperLedger Fabric (HF) framework. A collaborative network is configured in the platform as a permissioned blockchain in which nodes represent organizations. Figure 1 shows a general schema of the configuration of a collaborative network.

Nodes are classified based on their tasks. Peer nodes are responsible for validating the transactions that are triggered by the applications. Ordered nodes are responsible for the ordering of the transactions. The distributed ledger is accessible to all organizations.

Organizations are represented mainly by a peer node. Optionally, an organization can also have an ordering node. So more than one ordering node can be defined on the network. Another alternative is to have only one independent ordering node for the network. This last alternative is the most used configuration.

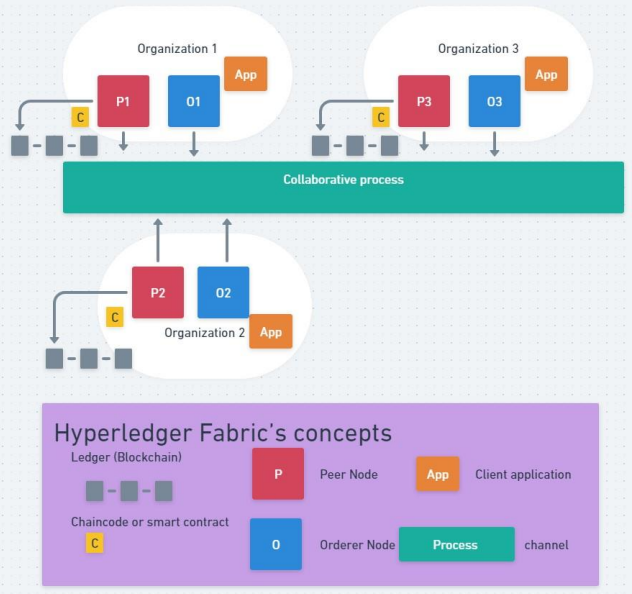

Figure 1. Generic Architecture of the Blockchain platform for collaborative processes

In addition, organizations interact with the blockchain through their own applications. Two types of applications must be provided by each organization. A sender application for triggering the transactions, i.e. the sending of the messages that will be recorded on the distributed ledger. A recipient application for querying the ledger to extract the messages for which the organization is the recipient, with the purpose of processing the messages internally and continuing with the next transaction if required.

The execution of a collaborative process with their interactions (i.e. the message-ordered exchange between the parties) in the proposed platform is carried out as follows. A message to be sent by an organization is generated by its sender application which triggers a transaction. The transaction contains the message information to be stored in the distributed ledger: the sender, the recipient, the message type, and the business document (the information that is shared in the process). The transaction is validated through smart contracts that are deployed in peer nodes. Smart contracts validate if the message of the transaction is a valid message in terms of the expected ordering of the interactions defined in the logic of the collaborative process. Once the transaction is validated and the message is stored in the ledger, the peer nodes of the organizations are informed about the update of the ledger. Thus, the organization that is the recipient of the message queries the ledger to extract the message and its content with the purpose of processing it and executing its internal activity. Also, each organization can query the ledger to know if it is responsible for generating the next transaction to send the following message of the process. Thus, through the transactions it is registered on the ledger the message exchange of the collaborative processes, which guarantees the correct execution of the business logic agreed by the organizations.

In this platform, we propose the use of channels for executing the transactions related to collaborative processes. A channel is defined by each collaborative process to be executed. Because each channel contains a distributed ledger, it means that all the execution instances (cases) of a collaborative process are managed through the same channel, and the data is stored in the ledger that is specific to that process. Thus, instances of the different processes are managed independently, and data privacy is achieved at the level of the processes. Organizations that are involved in a process cannot access to data of other processes in which they are not involved.

Design and development of smart contracts for collaborative processes

For the execution of collaborative processes, a smart contract is required for each collaborative process. The smart contract should implement the validation logic that makes it possible to determine if a message delivery, based on a transaction triggered by an application belonging to one of the organizations that make up the network in the blockchain, is correct. This implies verifying that the format, content, and order of the message are correct with respect to what is defined by the organizations in a collaborative process model.

With HyperLedger Fabric, smart contracts are defined in chaincodes (packages of code) that are associated with a channel. So chaincodes contain the implementation of the business logic to be fulfilled about the choreography of interactions agreed on in the collaborative process. Thus, chaincodes execute verification logic that ensures the integrity and validity of the message ordering before the messages are stored in the ledger.

In HyperLedger Fabric, chaincodes can be developed in several languages such as Javascript, Java, Go, etc, through the use of specific libraries (such as "org.example.ledgerapi.State", "org.hyperledger .fabric.contract" for Java). Using these libraries, the data of a smart contract such as the name, the version, a description, and other data are defined.

The following code describes parts of a Java chaincode for a collaborative process, which specifies the information fields about the contract.

```
@Contract(
    name = "basic",
    info = @Info(
       title = "Order Management",
       description = "The logic of the collaborative
process 'Order Management'",
       version = "0.0.1-SNAPSHOT",
       license = @License(
                        name = "Apache 2.0 License",
url="http://www.apache.org/licenses/LICENSE2.0.html")
,       contact = @Contact(
        email = "info@collaborativenetwork.com",
        name = "Collaborative Network",
        url = "https://collaborativenetwork.com")))
       ……. . . .

       ……. . . .
```

Then, the logic of the contract is defined in a class that implements the ContractInterface interface, for instance:

```
public class OrderManagementContract implements
ContractService {
        ……
}
```

In this class are implemented the methods to execute and validate the transactions that are triggered by the organizations on the corresponding channels. The methods are specific to each collaborative process and they check the conditions that should fulfill a transaction to be valid. These methods are annotated with `@Transaction`. In the proposed platform, to validate the transaction that implies the sending of message, the contract requires implementing a `sendMessage` method as follows:

```
public Message sendMessage(Context ctx, String
processId, String sender, String recipient, SpeechAct
messageType, BusinessDocumentSchema document) { {
        ……


    /*** It specifies the creation of a message for a
    process instance, i.e. the interaction
    information that then will be stored in the
    ledger ***/


    Message processMessage = new ProcessMessage
    (processId, sender, recipient, messageType,
    document);

    /*** Here it should be included the logic that
    validates the message and the fulfillment of the
    business logic of the collaborative process ***/

    /*** In case the business logic is not fulfilled,
    here should include the logic to manage
    exceptions and send alert messages to reject the
    transactions, by using the exception managers
    provided by the Hyperledger API ***/

    /*** If the message validation is ok, the message
    is serialized in JSON format ***/
    String message= genson.serialize(processMessage);

    /*** Then, the message is saved along with the
    new state on the ledger ***/

    stub.putStringState(processId, messageJson);

    return processMessage;
}
```

Deployment of smart contracts for implementing collaborative processes

To validate transactions in the blockchain for a particular collaborative process, organizations must install a chaincode (smart contract) in their Peer nodes that are attached to the channel that is associated with that collaborative process. In HF, a chaincode is implemented in a channel using a process called Fabric chaincode lifecycle whereby multiple organizations are allowed to agree on how a chaincode will be operated before it is used to accept transactions. Once the package code of the chaincode is developed, the implementation procedure of the chaincode is as follows. First, the organizations should

install the chaincode package in their Peer node. Then, the chaincode is settled to the corresponding channel.

Validation process of a transaction and the update of the ledger

The transaction workflow, i.e. the procedure of receiving a transaction triggered by the sender application of one of the organizations participating in the network until the update of the ledger with the output of the transaction consists mainly of three stages: execution, ordering, and validation.

1. Execution

The sender application of an organization, that wants to send a message, initiates a request to evaluate a transaction proposal for the sendMessage transaction type. The peer node of the organization executes the transaction by invoking the sendMessage method of the smart contract, with the input parameters provided by the client application. This transaction is run without any effect on the ledger. The peer then returns its transaction result to the sender application. Also, the transaction proposal is forwarded to the required endorsing peers which also execute the transaction and return their results to the initial peer. If most of these nodes accept it, they return the transaction signed and encrypted. The responses are collected, and if they collectively satisfy the endorsement policies, it forwards the transaction to the ordering service (running on the orderer nodes).

We defined that all the peer nodes of the channel are endorsing peers of the transactions, which implies that all the peers will execute the transactions to check the order of the message. Also, the format and content of the message are also checked. The output of the transaction is the serialized message that will be stored in the ledger.

2. Ordering

The ordering service receives transactions containing signed and endorsed proposal responses from one or more applications (because several transactions corresponding to different process instances can occur concurrently), and orders and packages the transactions into blocks. These are the blocks (which are also ordered) consisting of endorsed and ordered transactions that make up a ledger. The blocks are then saved to the orderer's ledger and prepared to be distributed to all peers on the channel.

3. Validation and commitment

The third phase of the transaction workflow is the distribution of the ordered transactions from orderer to peer nodes. This implies that when a new block is generated by the ordering service, all the peers connected to the channel receive a copy of the new block. Each peer processes the block independently but in the same way as every other peer on the channel. This ensures consistency on the ledger. Upon receipt of a block, a peer processes each transaction in the sequence specified in the block. For each transaction, the peer verifies that the transaction has been endorsed by the required organizations according to the endorsement policies for the chaincode that generated the transaction. In this platform, the endorsement policy used for the chaincode is that all organizations (peers) must endorse the transaction, so all the parties know and validate what is registered as an interaction of the collaborative process. After a peer has successfully validated each individual transaction, it updates the ledger. Thus, after all peers execute it, the local ledgers will have the same result of blocks and world state.

Finally, when a block is committed to a peer's ledger, that peer generates an event. Block events include the full block content. Chaincode events that the chaincode execution has produced can also be published at this time. The recipient applications of the peers are registered for notification of these event types. The event notification concludes the third and final phase of the transaction workflow.

These events are processed by the recipient applications of the peers. Each peer evaluates if the block added to the ledger contains messages in which the peer is the recipient. If that occurs, the recipient application can invoke an internal application of the organization associated with the peer for processing the content of the message. As a result, the organization can determine if it is required to send a message as part of the logic of the process. If needed, the organization will have to trigger another transaction and a new transaction workflow will be executed for the next message of the process.

Case Study

In this section, we illustrate the functionality of the proposed blockchain platform with a case study from the supply chain management domain. The Collaborative Order Forecast collaborative process is implemented. Figure 2 shows the conceptual model of this process that was defined as an interaction protocol with the UP-ColBPIP language [17]. The goal of this process is for participants to agree on a short/medium-term order forecast. The process begins with the retailer who sends a request for an order forecast to the manufacturer. The OrderForecastRequest business document contains the information about the requested order forecast, such as the time-horizon, the products, and so on. The manufacturer processes the request, evaluates it, and responds with one of the two alternatives: (1) sends an agree response with the "agree" message containing the OrderForecastRequestResponse business document, or (2) sends a reject response with the "refuse" message containing the OrderForecastRequestResponde business document. In this last case, the process ends.

If the manufacturer agrees on generating and order forecast based on the request, then the retailer must send independently (in a parallel way) two messages. One is the "inform" message that conveys the POSForecast business document, which contains the sale forecasts of all its

points of sale. The other one is the "inform" message that conveys the PlannedEvents business document, which contains the promotions and sales strategies.

With the received information, the manufacturer generates internally an order forecast (OrderForecast business document) that then is sent with an "inform" message to the retailer. Finally, the process ends.

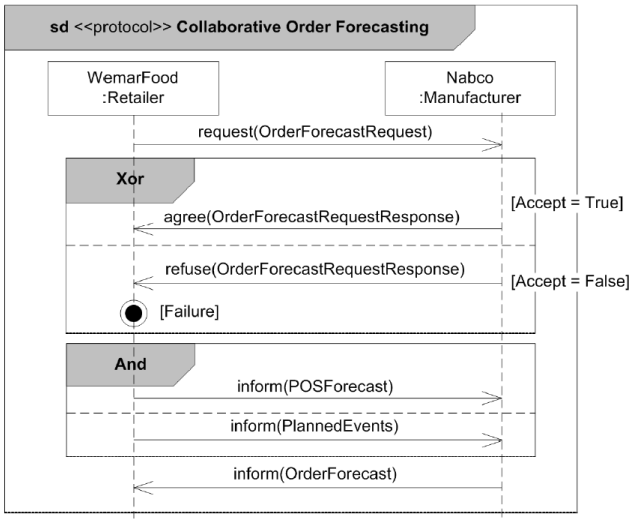

Figure 2. Collaborative Order Forecast process model

Definition of the blockchain network with Hyperledger Fabric

The configured blockchain network is defined by the following components:

- A peer node for the retailer organization and another one for the manufacturer
- An orderer node that is independent of the organizations is configured for the network.
- A channel with the name of the collaborative process is defined and both peer nodes are associated with it. Thus, the ledger of the channel stores just information about execution instances of this process. The communication of the organizations (peer nodes) is managed through the execution of transactions on this channel.
- A chaincode was developed for this process and it was deployed and linked to both peer nodes. The chaincode contains the transaction methods for each of the messages defined in the collaborative process model. The methods validate that the transaction (the message to be sent, i.e. the message to be stored in the ledger) is valid based on the status of the blocks in the ledger.
- Each organization has two applications for managing this process, the sender and the recipient applications. The Sender application is used to trigger transaction proposals that correspond to the transactions that represent the sending of messages by a peer. The receiver application listens for update events on the ledger, and contains the logic to process the messages that are extracted from the ledger, i.e. for the messages in which the organization is the recipient.

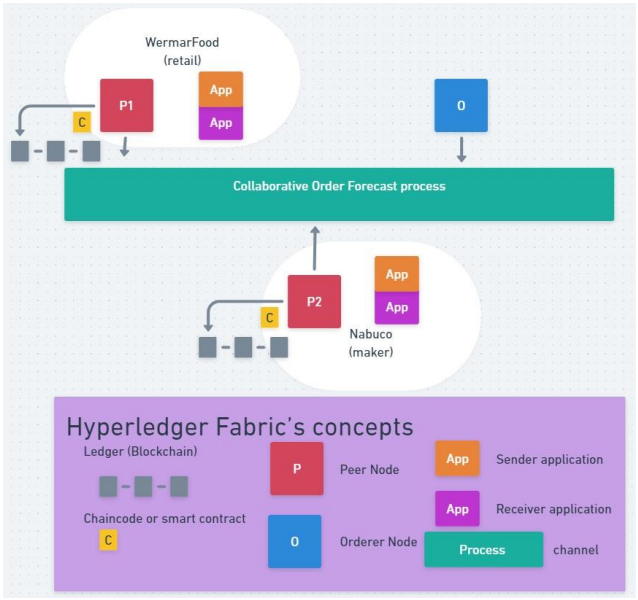

Figure 3. Configured blockchain network of the Collaborative Order Forecast process

Definition of smart contracts and chaincodes

For both organizations can make transactions on the channel of the process and store the messages on the ledger, smart contracts must be developed to define how these transactions will be. The smart contracts contain the rules that govern the control flow of the interactions (the message exchange) among the organizations.

In Hyperledger Fabric it is often to use the terms smart contract and chaincode interchangeably, however, they are different. A smart contract defines the transaction logic that controls the lifecycle of an object contained in the world state. A chaincode contains a package of smart contracts which is then deployed to a blockchain network. Thus, smart contracts contain the logic that governs the transactions, whereas chaincode governs how smart contracts are packaged for deployment.

For the channel defined on the network, we deployed a chaincode that contains the CollaborariveOrderForecast smart contract we developed for the transactions of the collaborative process. Figure 4 shows part of the Java code of it. This smart contract contains the methods that implement the transactions that can occur in the blockchain. The implementation of each method contains the logic that checks that a transaction is valid in terms of the message ordering to be fulfilled for the process. Thus, for each of the messages defined in the collaborative process, a transaction method was implemented.

```
public class CollaborativeOrderForecast implements ContractService {

    public Message requestOrderForecast(Context ctx, String processId,
            String sender, String recipient,
            OrderForecastRequest document) {
        ......
    }

    public Message agreeOrderForecast(Context ctx, String processId,
            String sender, String recipient,
            OrderForecastRequestResponse document) {
        ......
    }

    public Message refuseOrderForecast(Context ctx, String processId,
            String sender, String recipient,
            OrderForecastRequestResponse document) {
        ......
    }

    public Message informPosForecast(Context ctx, String processId,
            String sender, String recipient,
            POSForecast document) {
        ......
    }

    public Message informPlannedEvents(Context ctx, String processId,
            String sender, String recipient,
            PlannedEvents document) {
        ......
    }

    public Message informOrderForecast(Context ctx, String processId,
            String sender, String recipient,
            OrderForecast document) {
        ......
    }
}
```

Figure 4. Smart contract implementation for the Collaborative Order Forecast process

Every smart contract has an endorsement policy associated with it. This endorsement policy identifies which peers must approve transactions generated by the smart contract before those transactions can be identified as valid. For the chaincode we deployed in the channel, an endorsement policy was defined that specifies both peers of the channel must approve all the transactions.

Figure 5 shows the logic for the method that implements the transaction for sending the "agree" message of the process. First, the message entity to be saved on the ledger is generated. Then the last message that was registered in the ledger for the process instance of interest is fetched. The function getLastMessageFromLedger contains the logic that consults the blocks and the world state of the ledger to obtain the last registered message of the process instance. Then, the conditions of the "agree" message are checked. In this case, the conditions for a valid transaction are the following: the previous message must correspond to the "request" message, the type of the document must be OrderForecastRequest, the sender must fulfill the role of Retailer, and the recipient must fulfill the role of Manufacturer. Finally, the message is converted to JSON format and put into the ledger.

```
public Message agreeOrderForecast(Context ctx, String processId,
        String sender, String recipient,
        OrderForecastRequestResponse document) {
    /*** It creates the message of the received process instance ***/
    MessageType = SpeechAct.AGREE;
    Message newMessage = new ProcessMessage (processId, sender,
                                recipient, messageType, document);

    /*** This queries the ledger for the last transaction of the
        processId and the last executed message is obtained ***/
    Message lastMessage = getLastMessageFromLedger(processId);

    /*** This check if the message is valid against the message
         ordering of the process ***/
    if (lastMessage.getType().equals(SpeechAct.REQUEST)
        && lastMessage.getDocument() instanceof OrderForecastRequest
        && newMessage.getSender().equals(OrganizationRole.RETAILER)
        && newMessage.getRecipient().equals(OrganizationRole.MANUFACTURER)) {
        /** the transaction is valid and the message is saved on the ledger *
        String messageJson = genson.serialize(newMessage);
        stub.putStringState(processId, messageJson);
        return processMessage;
    } else {
        /** the transaction is not valid. An exception can be generated **/
        ....
        return null;
}
```

Figure 5. Implementation of the transaction method for the "agree" message

The rest of the methods of this smart contract have the same structure in their logic. The conditions in terms of the message ordering that were defined in the different transaction methods are:

- A "request" message can be sent if there is not a previous message for the process instance.
- "agree" and "refuse" messages only can be sent if the last message is the "request" message.
- POSForecast and PlannedEvent "inform" messages can be sent only if the last message is the "agree" message.
- OrderForecast "inform" can be sent only if POSForecast and PlannedEvents were sent before.
- For all the methods: only one occurrence of the message for the process instance can be present on the ledger.

Execution and Ledger status

As an example of the execution of an instance of the process, Figure 6 shows the status of the ledger after the end of the execution of this instance. Five transactions were executed, and they were put into two blocks (1 and 2). The world state shows the last three executed transactions. The key identifies the process instance and the message of that process instance. The string of the key has this format: process ID + "." + message type + "." + document abbreviation. For the executed instance, its process ID is "PI1".

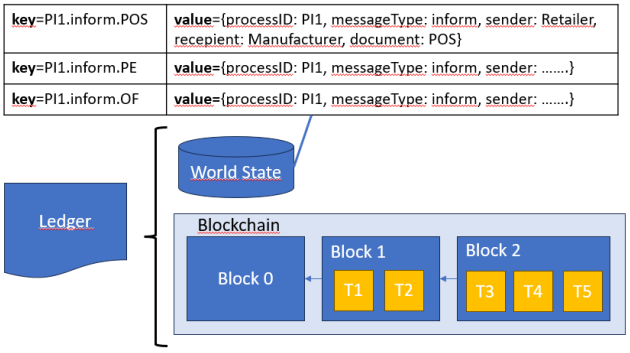

Figure 6. Ledger status after the execution of the process instance PI1

We can also see that the blockchain contains three blocks. Block 0 is the genesis block, it does not contain any transactions. Block 1 contains transactions T1 and T2. T1 corresponds to the "request" message and T2 corresponds to the "agree" message. The rest of the transactions in block 3 correspond to the transactions that are in the world state. Thus, all the results of the transactions in terms of the message sent from one organization to the other are stored in the ledger.

Related Works

There are several works that were proposed to take advantage of the blockchain features for collaborative processes. However, the way that the blockchain is implemented in these works is different from the approach we propose in this paper.

Caterpillar [13] is an open source tool created for collaborative business process management that runs on the Ethereum blockchain. This architecture has an execution layer within the chain intended to execute the logic of the business process in the form of a set of smart contracts. Smart contracts are generated by Caterpillar from BPMN model inputs to handle the control-flow of the process model. Other Smart contracts handle the interaction with applications external to the blockchain and validate the data entered. Smart contracts act as mediators for forwarding a request and receiving the corresponding response. All smart contracts are coded in the Solidity language. The state of each instance of a business process is maintained on the blockchain and the routing of workflows is carried out through Smart contracts generated by a BPMN to Solidity compiler.

Each smart contract encapsulates the workflow routing logic of the business process model, specifically containing variables to encode the state of the business process instance and scripts to update this state each time a task is completed, or an event occurs. The events generated by the smart contracts are recorded in the blockchain in a log of the platform, which is accessible from outside the chain, notifying the external component of the chain of an event that has occurred.

Because Caterpillar run on a non-permissioned blockchain as Ethereum by using the PoW consensus protocol, the cost to execute transactions is high in terms of resources (for mining) and performance of the transactions. Our approach is based on a permissioned blockchain as Hyperledger Fabric, which allows to reduce these implementation costs. In addition, we propose to store in the blockchain only the messages issued between the parties as the state of the execution of the corresponding instance of the business process that is executed, whose type, order, correct sender, and receiver are controlled by the smart contracts. Thus, there is no kind of centralized component to govern the global state of the instances of the collaborative processes.

Lorikeet [3] is a tool that consists of a modeling user interface, a BPMN translator, a log generator, and a blockchain enabler. The BPMN translator converts the BPMN model into a Solidity smart contract considering record models, for execution on Ethereum. The ethtrigger communicates with the Ethereum blockchain node and handles smart contract deployment, execution, and interaction.

In Lorikeet, it is possible to define an access control policy that regulates registration operations. This means the ability to restrict the invocation of actions and access to assets. Additionally, it is possible for process instances to manipulate log entries. However, this is a limitation of Lorikeet regarding privacy. It is not possible to have organizations or detailed confidentiality. It is possible to define which function a business process step is supposed to execute, but confidentiality cannot be guaranteed. Lorikeet is a tool intended to translate business processes into smart contracts on a public blockchain. This does not focus on executing and/or monitoring collaborative business processes.

CoBuP [14] is a multi-layer architecture proposed to monitor the execution of collaborative business processes using smart contracts. To improve flexibility in the logical structure of smart contracts proposed by other authors, the architecture considers three smart contracts that are generated in the different layers internal to the blockchain. In the first layer, they generate a generic smart contract, Interpreter, which includes variables and functions to encode the structure of any BPMN model. This smart contract is generated only once and is responsible for generating the ProcessInstance and ResourceInstance smart contracts. The first statically encodes the data structure of any business process model and includes the BPMN model workflow. Each business process is translated into a ProcessInstance smart contract in XML. This contract has a list of elements that define the business process (task, event, xor/or/and gateways, etc.), policy associated with its execution, and the state of the element (Create, Ready, etc.). A ProcessIntance contract contains the address in the blockchain of the associated ResourceInstance smart contract, where the participants in the collaborative process are listed, and their role within it, enabling the participant to execute elements of the process defined in the ProcessInstance contract. Collaborative business process participants agree to query the state of the process defined in the ProcessInstance contract where the state of each element is updated when a participant executes a process element.

The CoBuP architecture has been tested on a public Ethereum blockchain platform, where the smart contracts were coded using Solidity. The authors maintain that it is carried out on this platform to make the corresponding efficiency comparisons, but that it could be implemented in a permissioned network like Hyperledger Fabric. However, no works were reported with Hyperledger

Fabric. CoBuP monitors the control flow that defines the collaborative business process, while the blockchain architecture proposed in this work carries out the execution and control by recording the sequence of messages between the parties in the blockchain. In addition, a private blockchain platform based on HF is defined, which guarantees greater reliability between participants in the collaborative business process.

Conclusions and Future Work

In this work, we proposed a blockchain platform to execute collaborative business processes. This platform enables the fulfillment of the requirements for executing collaborative processes: autonomy of the organizations, decentralization, peer-to-peer interactions, privacy of the shared and private information, trust, and execution of the agreed interactions.

The platform is based on Hyperledger Fabric with the purpose of providing a permissioned blockchain for collaborative processes. The goal is to take advantage of the trust and privacy mechanisms provided by blockchain technology, without incurring low performance in the transactions and high hardware resource costs to execute transactions for the exchange of messages between the organizations.

The use of channels for implementing collaborative processes on the blockchain allows for improving the privacy aspect of the data that organizations share. In this platform, several processes can be implemented by defining a channel for each of them. Thus, the organizations through their peer nodes are associated just with the channels (processes) in which they are involved, and they can only see the data of these processes.

A key element in the proposed blockchain platform is the use of smart contracts, which are deployed in chaincodes, for executing the logic of the collaborative processes. In this platform, the smart contracts have two main responsibilities:

- Ensure the business rules and logic defined in collaborative business processes are fulfilled and respected, in terms of the message types and message ordering.
- Generate the messages that are stored in the ledger with the right structure and information. Each smart contract implements the transaction methods for executing the sending of the process's messages.

In addition, the applied endorsement policy for smart contracts is that all the peers (organizations) must validate and sign the transactions, with the aim of achieving a high level of transparency and trust among the parties.

The proposed approach for smart contracts provides a blockchain completely decentralized. There is no communication with an external app or an internal component of the blockchain that governs the instances of the processes. All the organizations can query the status of the process instances by querying the ledger, and the logic of the message ordering is governed by the transactions (implemented in the smart contract of the process) that are signed and endorsed by the organizations.

Future work is about defining a method and a tool to generate the configuration of the blockchain and the code of the smart contracts from conceptual collaborative process models, by applying model-driven development principles and tools, as was implemented in previous works [12][16] for platforms based on other technologies.

References

[1] Androulaki,E, Barger,A, Bortnikov,V, Cachin,C, Christidis,A, De Caro,A, Enyeart,D, Ferris, C, Laventman, G, Manevich, Y. (2018). Hyperledger Fabric: a distributed operating system for permissioned blockchain. Proceedings of the Thirteenth EuroSysConference ACM pp. 30.

[2] Astigarraga, T., Chen, X., Chen, Y., Gu, J., Hull, R., Jiao, L., Li, Y., and Novotny, P. Empowering Business-Level Blockchain Users with a Rules Framework for Smart Contracts. In Proc. of International Conference on Service Oriented Computing (ICSOC). Springer, Cham, 2018, pp. 111–128.

[3] Binh Tran, A, Lu, Q, Weber,I. Lorikeet. A model-driven engineering tool for blockchain-based business process execution and asset management. Proceedings of the Dissertation Award and Demostration Insustrial Track at BPM, 2018.

[4] Cocconi, D., Roa, J., Villarreal, P. Collaborative Business Process Management Through a Platform Based on Cloud Computing". CLEI Electronic Journal, 21,2, 2018

[5] Cocconi, D., Villarreal, P. Microservices-based Approach for a Collaborative Business Process Management Cloud Platform. XLVI Latin American Computing Conference (CLEI), Loja, Ecuador, October 19-23 2020.

[6] Di Ciccio, C; Cecconi, A; Dumas, M; García-Bañuelos, L;Lopez-Pintado, O; Lu,Q; Ponomarev,A; Trans,A; Weber,I (2019)Blockchain Support for Collaborative Business Processes, Informatik Spektrum Vol.42, pp182-190, 2019.

[7] García-Bañuelos, L., Ponomarev, A., Dumas, M. and Weber, I. Optimized execution of business processes on blockchain. Lecture Notes in Computer Science, Cham:Springer, vol. 10445, 2017.

[8] Henry, T., Laga, N, Hatin, J., Gaaloul, W., Boughzala, I.. Cross-collaboration processes based on blockchain and IoT: a survey. HICSS 2021: 54th Hawaii International Conference on System Sciences, Jan 2021, Maui, Hawaii, United States. pp.4291-4300

[9] Hiroaki Nakamura, Kohtaroh Miyamoto, and Michiharu Kudo. Interorganizational business processes managed by blockchain. In International Conference on Web Information Systems Engineering, pages 3{17. Springer, 2018.

[10] Hyperledger Fabric Documentation. https://hyperledger-fabric.readthedocs.io/en/release-2.5/ Accessed date 11/9/2023

[11] Iansiti M, Lakhani KR (2017) The truth about blockchain. Harvard Business Review 95, 1, pp.118–127

[12] Lazarte, I. M., Thom, L. H., Iochpe, C., Chiotti, O., & Villarreal, P. D. “A distributed repository for managing business process models in cross-organizational collaborations”. Computers in Industry, 64, 1, 2013, pp. 252-267.

[13] Lopez-Pintado,O, Garcia-Bañuelos,L, Dumas,M, Weber,I, Pnomarev,A. (2019). CARTERPILLAR: a business process execution engine on the Ethereum blockchain. Software: Practice and Experience,7,49, pp.1162-1193.

[14] Lounkil,F, Boukadi,K, Abed,M, Ghedira-Guega,C. (2021). Decentralized collaborative business process execution using blockchain. World Wide Web, 24, pp. 1645-1663.

[15] 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, Garc´ia-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.

[16] Tello-Leal, E., Chiotti, O., Villarreal, P.D.: Software agents for management dynamic inter-organizational collaborations. IEEE Latin America Transactions 12(2) (2014) 330-341.

[17] Villarreal, P., Lazarte, I., Roa, J., Chiotti, O.: A Modeling Approach for Collaborative Business Processes based on the UP-ColBPIP Language. In: Rinderle-Ma, S., Sadiq, S., y Leymann, F. (eds.) Business Process Management Workshops. 318-329 Springer Berlin Heidelberg, Berlin, Heidelberg (2010).

[18] Weber, I., Xu, X., Riveret, R., Governatori, G., Ponomarev A. and Mendling, J. Untrusted business process monitoring and execution using blockchain. Lecture Notes in Computer Science, Cham:Springer, vol. 9850, 2016.

[19] Weske, M.. Business process management: concepts, languages, architectures (2nd. Edition). Springer Publishing Company, 2012.

[20] Xian Rong Zheng & Yang Lu (2021) Blockchain technology – recent research and future trend, Enterprise Information Systems, DOI: 10.1080/17517575.2021.1939895.

[21] Yang,L, Wakefield,R, Lyu,S, Jayasuriya,S, Han,F, Yi,X, Yang,X, Amarasinghe,G, Chen,S. Public and private blockchain in construction business process and information integration. Automation in Construction. 118,2

[22] Zheng,Z, Xie,S, Dai, H-N,Chen, X,Wang, H, (2018) Blockchain challenges and opportunities: a survey International Journal of Web and Grid Services, 14, 4, pp.352-375,