# ChorChain: A Model-Driven Framework for Choreography-Based Systems Using Blockchain*

Flavio Corradini[1], Alessandro Marcelletti[1], Andrea Morichetta[1], Andrea Polini[1], Barbara Re[1], and Francesco Tiezzi[1]

University of Camerino, Camerino 62032, IT
{name.surname}@unicam.it

**Abstract.** The complexity in the development of distributed systems has increased the necessity to consider new model-driven methodologies for their implementation. This complexity is higher when combined with the lack of a trusted execution environment necessary to guarantee the correct behaviour of all the involved participants. In such context, the BPMN standard, in particular the choreography diagram, is one of the advocated modelling languages able to represent the interactions that should occur among distributed components. This modelling language, combined with the immutable and trusted nature of the blockchain technology, provides a promising solution to master complexity in developing and executing trusted distributed systems. This paper describes a model-driven methodology based on blockchain technology and the related framework named ChorChain. Starting from a BPMN choreography model, ChorChain generates the corresponding smart contract and the respective infrastructure for the choreography execution. To show the ChorChain feasibility, we have been tested it on a Room Booking scenario.

**Keywords:** Distributed systems · BPMN · Choreography · Blockchain

## 1 Introduction

In the last years, model-driven engineering methodologies are used to master the complexity of coordination in distributed systems involving multi-party organisations. These methodologies allow the passage from abstract modelling languages, describing the interaction logic, to code executable on a targeted run-time environment. In such a direction, the BPMN standard [1] nowadays has become a prominent modelling language to describe collaborative systems in distributed environments [4, 5, 10]. In this paper, we rely on *choreography diagrams*, which permit us to describe system interactions in terms of the exchange of messages from a global perspective, without exposing the internal behaviour of each component.

* Copyright © 2021 for this paper by its authors. Use permitted under Creative Commons License Attribution 4.0 International (CC BY 4.0).

[1] https://www.omg.org/spec/BPMN/2.0/PDF

In addition to coordination complexity, distributed systems may lack of trust among the involved parties. This trustelessness issue [9] can be mitigated by the inclusion of blockchain technology, as envisioned in [6]. The blockchain, indeed, guarantees the integrity and the immutability of data, without relying on a central authority or any particular participant [1, 7, 8]. This allows choreography participants to have a clear view of the ongoing system execution and tangible proofs of the actions performed by the counterparts [3, 2].

In this paper, we focus on the use of the model-driven methodology and the ChorChain framework, introduced in [3], for the development of trustable choreography-based systems. In doing this, we focus on a case study related to a room booking system. The proposed approach supports the full life-cycle of choreographies, from their modelling to their publishing and instantiation, until their deployment and execution in the Ethereum blockchain. In particular, the deployment phase is supported by ChorChain thanks to the automatic generation of Solidity smart contracts. These contracts indeed are specifically generated to implement the choreography workflow, thus forcing the correct behaviour of each participant.

The rest of the paper is organised as follow. Section 2 describes the proposed methodology, while Section 3 shows the ChorChain framework in practice applied to a case study. Finally, Section 4 concludes the paper.

## 2 ChorChain Methodology

In this section, we describe the methodological aspects of the proposed approach that is divided into three main phases in Fig. 1 (i) system modelling, (ii) instantiation and (iii) execution.

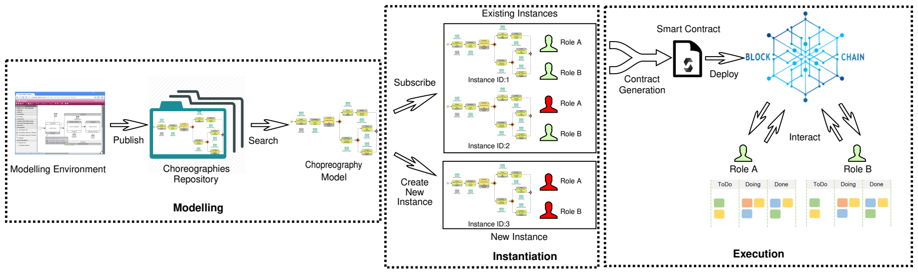


**Fig. 1.** ChorChain methodology.

The first phase consists of the modelling of a distributed system by means of a choreography specification. This permits focusing on the interactions, abstracting from low-level implementation details and reducing the development effort. The created model is published in a choreography repository making it publicly available as a blueprint.

A model can be used to generate multiple instances with the same structure but with different participants. When a choreography instance is created, in order to be executed it has to be subscribed by the participants that are willing to play a specific role.

Once all the required roles are filled, the corresponding smart contract is generated and then deployed, automatically, on the blockchain. At this point, the execution starts, and the participants can cooperate following the message protocol established by the choreography specification, and implemented in the smart contract.

## 3 ChorChain on Room Booking Case Study

In this section, we show how the proposed methodology was concretely applied on a realistic example. For this purpose, we demonstrate the ChorChain functionalities, with the Room Booking case study. The interested reader can visit *http://pros.unicam.it/chorchain/* for more details about the tool and its usage. The model shown in Fig. 2 describes the interactions between a *Client* and a

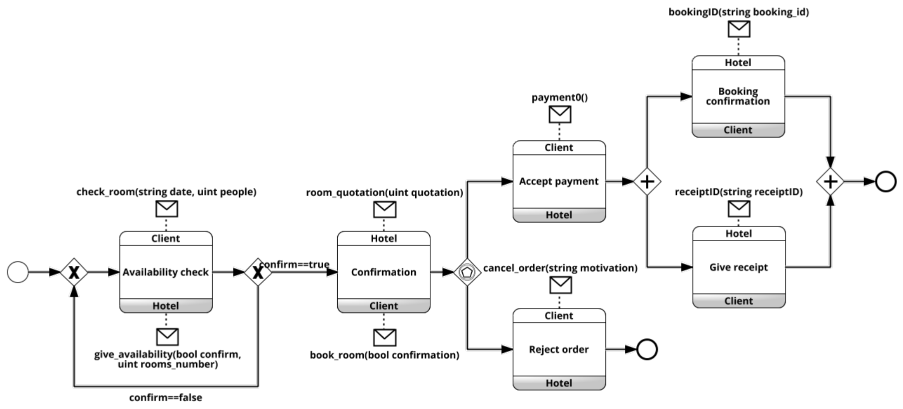
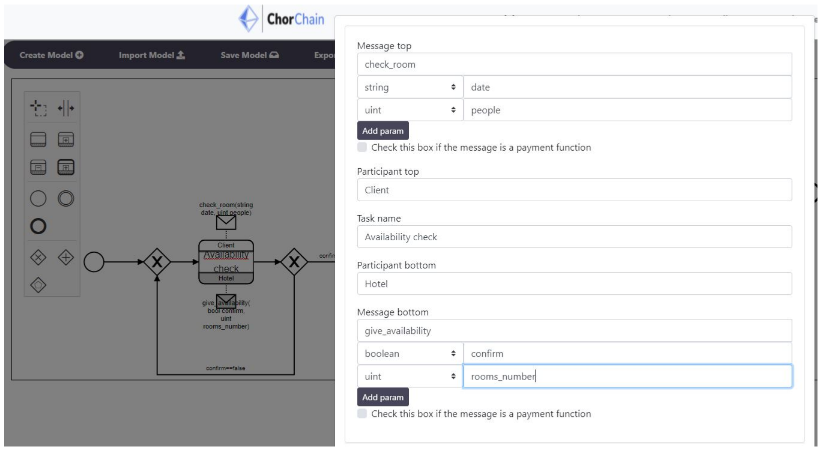

bookingID(string booking_id)
Hotel
Booking confirmation
Client
payment0()
Client
Accept payment
Hotel
receiptID(string receiptID)
Hotel
Give receipt
Client
check_room(string date, uint people)
Client
Availability check
Hotel
give_availability(bool confirm, uint rooms_number)
confirm==false
confirm==true
room_quotation(uint quotation)
Hotel
Confirmation
Client
book_room(bool confirmation)
cancel_order(string motivation)
Client
Reject order
Hotel

**Fig. 2.** Room booking case study.

*Hotel* for the booking of a room. In particular, the process starts with the request of availability for a room on a specific date and a given number of guests. In absence of a free room, the client iterates making new requests. In the positive case, instead, the hotel replies with the room number and its quotation. This open two possible subsequent actions for the client: (i) the rejection of the order that makes the process ends or (ii) the payment activity of the booked room. After the payment, the hotel provides in parallel the booking ID and the receipt ID.

Here below we show the steps that the parties need to accomplish to collaborate on such a scenario using the ChorChain framework. The first required step is the authentication. The parties need to sign-up in the system by registering their Ethereum accounts. This is necessary to associate all the operations performed in the blockchain with a participant. Once this step is completed, the sign-in can be done through the Metamask plugin that facilitates the user in managing the Ethereum account.

Then, the participant can start the definition of the room booking model through the integrated modelling environment exploiting the choreography elements. Additional information related to (i) messages and (ii) guards is also necessary. These annotations are facilitated by a dedicated interface accessible for each choreography task. Fig. 3 shows the ChorChain interface for the definition of the messages and parameters in the *Availability check* task. The result of this procedure is a list of parameters included between brackets after the message name; e.g. *check_room(string date, uint people)* defines the first message sent by the client. This format is necessary for performing the underlying function call in the generated smart contract. Similarly, the guards annotation in the outgoing sequence flows of exclusive gateways is used to enforce the control flow of the smart contract. This will be translated later as conditions automatically verified in the contract, guaranteeing a proper execution of the choreography model when deployed in the blockchain.



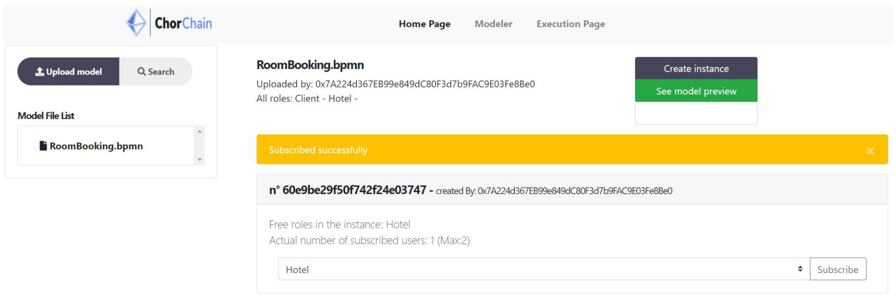
Message top
check_room
string | date
uint | people
Add param
Check this box if the message is a payment function
Participant top
Client
Task name
Availability check
Participant bottom
Hotel
Message bottom
give_availability
boolean | confirm
uint | rooms_number
Add param
Check this box if the message is a payment function

**Fig. 3.** ChorChain modeller panel.

Once the Room Booking model is completed, it is saved in the ChorChain repository, and it is shown in the model file list on the home page (Fig. 4) accessible by every logged-in user.

Selecting the *RoomBooking.bpmn* file, ChorChain shows the owner of the model, the maximum number of involved participants and the required roles. Furthermore, the graphical preview and the instantiation functionalities are accessible by clicking on the related buttons (*Create instance* and *See model preview*). In the specific case of Fig. 4, the Room Booking instance contains the address of the creator, and the two roles defined in the model: the Hotel and the Client. The created instance will remain in a "suspended" state until all the mandatory roles are subscribed. To fill these roles, participants that are interested to participate as client or hotel can subscribe and associate their Ethereum accounts to the role. When all roles are fulfilled and no more vacant roles exist, ChorChain considers the partnership complete and starts the generation and deployment of the Solidity smart contract.

The code generation is completely automatic and transparent to the involved participants that have no knowledge about low-level implementation details. In the generated smart contract, messages are represented by public functions that can be invoked only by a specific participant. For gateways, instead, the



Model File List

RoomBooking.bpmn

RoomBooking.bpmn

Uploaded by: 0x7A224d367EB99e849dC80F3d7b9FAC9E03Fe8Be0

All roles: Client - Hotel -

Subscribed successfully

n° 60e9be29f50f742f24e03747 - created By: 0x7A224d367EB99e849dC80F3d7b9FAC9E03Fe8Be0

Free roles in the instance: Hotel

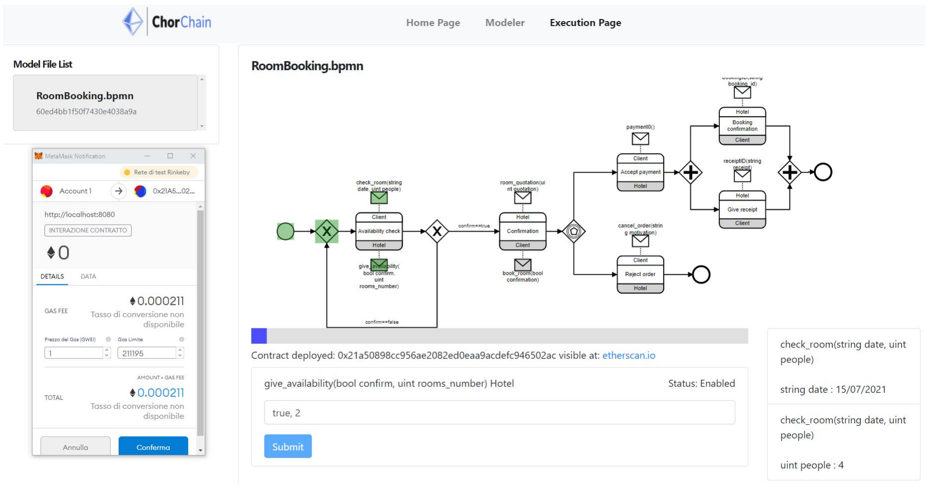
Actual number of subscribed users: 1 (Max:2)

Hotel

**Fig. 4.** ChorChain instance and subscription view.

related functions are automatically triggered internally by the smart contract. The interested reader can refer to *http://pros.unicam.it/RoomBooking.sol* for more details about the generated Room Booking smart contract.

Once the smart contract of the Room Booking scenario is deployed, it is accessible by the parties in the ChorChain execution page (Fig. 5). Here both the hotel and the client can interact with each other following the deployed choreography specification. In particular, the left-hand side of the interface reports a list



Model File List

RoomBooking.bpmn
60ed4bb1f50f7430e4038a9a

http://localhost:8080

INTERAZIONE CONTRATTO

0

GAS FEE 0.000211
Tasso di conversione non disponibile

Prezzo del Gas (GWEI) 1

Gas Limite 211195

AMOUNT + GAS FEE

TOTAL 0.000211
Tasso di conversione non disponibile

RoomBooking.bpmn

Contract deployed: 0x21a50898cc956ae2082ed0eaa9acdefc946502ac visible at: etherscan.io

give_availability(bool confirm, uint rooms_number) Hotel Status: Enabled

true, 2

check_room(string date, uint people)

string date : 15/07/2021

check_room(string date, uint people)

uint people : 4

**Fig. 5.** ChorChain execution page with hotel role sending the confirmation transaction.

of all the deployed contracts, to which the participant is subscribed. In the centre, it is shown a preview of the model with its current execution state highlighted by green messages. For the active messages, a form is dynamically constructed below the model, to insert the parameters and executing it (*give_availability*). Notably, the submission form is shown only to the participant subscribed and in charge to play that role. The counterpart (client in the *give_availability* message) has only knowledge of the active message, without the possibility of submitting it. For each message submission, ChorChain produces a blockchain transaction that has to be confirmed through the Metamask pop-up. As soon as the transaction is stored in the blockchain, an event is emitted and is used for notifying all the other participants about the advancement of the system status until an end event is reached. To log the current value of variables, on the bottom-right side of Fig. 5 a panel is shown. In particular it contains the *check room* message with its two parameters (*date* and *people*) with the respective values retrieved from the blockchain.

Considering costs for executing the Room Booking scenario we have 4,510,860 units of gas used for the deployment of the contract and 817,389 for its execution. On average each transaction uses around 116,769 units of gas. The execution time instead is strictly related to the technology used and we registered on average 15 seconds for the inclusion of each transaction in the Rinkeby network.

## 4 Conclusions

In this work, we faced the problem of the trustworthy implementation of distributed systems through the use of multi-party choreographies. The concrete realisations of such systems suffer from the lack of a concrete implementation that guarantees also trust to the participants. To address such a challenge we proposed in [3] a novel model-driven methodology for managing the whole choreography life-cycle, relying on a blockchain infrastructure. The methodology is concretely supported by the ChorChain framework, which implements all the functionalities from the diagram specification to its execution on the blockchain. We demonstrate here the feasibility of the solution and its effectiveness by applying the ChorChain approach to a Room Booking case study.

## References

1. Almeida, S., Albuquerque, A., Silva, A.: An approach to develop software that uses blockchain. In: Computer Science Conference. pp. 346–355. Springer (2018)
2. Corradini, F., Marcelletti, A., Morichetta, A., Polini, A., Re, B., Scala, E., et al.: Model-driven engineering for multi-party business processes on multiple blockchains. Blockchain: Research and Applications p. 100018 (2021)
3. Corradini, F., Marcelletti, A., Morichetta, A., Polini, A., Re, B., Tiezzi, F.: Engineering trustable choreography-based systems using blockchain. In: 35th Annual ACM SAC. pp. 1470–1479 (2020)
4. Di Ciccio, C., Cecconi, A., Dumas, M., García-Bañuelos, L., López-Pintado, O., Lu, Q., Mendling, J., Ponomarev, A., Tran, A.B., Weber, I.: Blockchain support for collaborative business processes. Informatik Spektrum **42**(3), 182–190 (2019)
5. López-Pintado, O., García-Bañuelos, L., Dumas, M., Weber, I., Ponomarev, A.: Caterpillar: a business process execution engine on the ethereum blockchain. Software: Practice and Experience **49**(7), 1162–1193 (2019)
6. Mendling, J., Weber, I., Aalst, W.V.D., Brocke, J.V., Cabanillas, C., Daniel, F., Debois, S., Ciccio, C.D., Dumas, M., Dustdar, S., et al.: Blockchains for business process management-challenges and opportunities. TMIS **9**(1), 1–16 (2018)
7. Porru, S., Pinna, A., Marchesi, M., Tonelli, R.: Blockchain-oriented software engineering: challenges and new directions. In: ICSE-C. pp. 169–171. IEEE (2017)
8. Rocha, H., Ducasse, S.: Preliminary steps towards modeling blockchain oriented software. In: WETSEB. pp. 52–57. IEEE (2018)

9. Tai, S.: Continuous, trustless, and fair: Changing priorities in services computing. In: Conf. on Service-Oriented and Cloud Computing. pp. 205–210. Springer (2016)
10. Tran, A.B., Lu, Q., Weber, I.: Lorikeet: A model-driven engineering tool for blockchain-based business process execution and asset management. In: BPM. pp. 56–60 (2018)