Non-Fungible Mutable Tokens for Flexible Choreography Roles and Ownership

Francesco Donini[1,2,3], Alessandro Marcelletti[1], Andrea Morichetta[1,*] and Andrea Polini[1]

[1]University of Camerino, Via Madonna delle Carceri 7, Camerino, 62032, Italy
[2]University of Pisa, Largo Bruno Pontecorvo 3, Pisa, 56127, Italy
[3]Institute of Informatics and Telematics CNR of Pisa, Via Giuseppe Moruzzi 1, Pisa, 56124, Italy

Abstract

Inter-organizational business processes involve distributed participants collaborating to achieve shared goals. Blockchain technology has emerged to support the decentralized execution of such processes, ensuring transparent and immutable tracks of participant duties and their roles within the process. These aspects are indeed crucial in a process as they provide accountability and proof of compliance with standards and regulations. To represent inter-organizational processes, Business Process Model and Notation choreographies have been widely adopted, as they specify distributed interactions between business participants. However, while they have found in blockchain a prominent solution for their implementation and execution, current solutions overlook the certification of participants' roles and ownership. An additional challenge is posed by the dynamism of business scenarios, where the introduction of new legislation or stakeholders requires flexible process management. This work proposes a novel framework and architecture leveraging Non-Fungible Mutable Tokens (NMTs) to manage ownership of participants and their roles in choreographies. NMTs dynamically represent participants and processes, allowing runtime updates while ensuring ownership and reproducibility. A canteen scenario was used to evaluate the feasibility of this approach.

Keywords

Blockchain, NMT, Choreography, Ownership, NFT, BPMN

1. Introduction

Inter-organizational business processes define interactions among distributed participants, willing to collaborate to reach a common goal [1, 2, 3]. In this context, blockchain emerged as an enabling technology to support inter-organizational processes over their lifecycle [4]. Blockchain enables a decentralized execution of the interactions, by enforcing business logic and providing a secure proof of executed activities, enhancing the trustworthiness of the process. This allows for monitoring activities [5, 6], relying on blockchain as a certified source about the behavior of the process and its participants. Having a clear and certified assignment of roles and operations within a business process is indeed essential for ensuring accountability, traceability, and effective governance [7, 8]. Trusted evidence of the actors involved, their rights over specific tasks, or of the process itself, supports not only operational transparency but also compliance with legal and industry standards. Certified participants with attested responsibilities simplify audit activities, making them more efficient and reliable. Furthermore, as processes evolve, knowing precisely who owns each part of the process becomes critical for managing change, allocating resources, and maintaining continuity. Governance is a critical factor in the success of a process [9], and they are particularly challenging in inter-organizational contexts where there is usually no explicit agreement [10]. This complexity is even more due to the various factors affecting business scenarios, leading to high dynamism and demand for runtime changes. Indeed, internal or external factors could lead to new situations, such as new regulations or agreements, requiring new stakeholders to join the process or to change the party covering a specific role. In this setting is essential to support the transfer of rights and roles, with the consequent transfer of tasks and duties. Over the

DLT2025: 7th Distributed Ledger Technology Workshop, June, 12-14 2025 - Pizzo, Italy
*Corresponding author.
francesco.donini@unicam.it (F. Donini); alessand.marcelletti@unicam.it (A. Marcelletti); andrea.morichetta@unicam.it (A. Morichetta); andrea.polini@unicam.it (A. Polini)

years, choreographies have been widely adopted to specify inter-organizational processes thanks to their ability to represent the interactions between business participants without exposing their internal behavior. In particular, the Business Process Management and Notation (BPMN) [11] is one of the most established standards enabling the design of choreographies. These kinds of models have also been integrated by blockchain-based solutions, taking choreography as a process specification and encoding it into a smart contract [12, 13]. Specifically, current solutions mainly enforce choreographies by means of smart contracts, used to encode aspects such as involved users, data to exchange, activities to execute, and constraints.

However, despite the integration of blockchain for implementing choreographies, the comprehensive management of participants and their roles has not gained as much attention. Indeed, while these solutions include the notion of participants and actions they have to perform, there are no solutions specifically aimed at handling process or role certification. This lack is even more notable when dealing with runtime changes and the need for flexibility arises, advocating for update capabilities. It is therefore crucial to have a solution that permits the certification participants and their roles while dealing with the need for flexibility.

To address these needs, in this work we propose the ChorNMT architecture to certify choreography participants and their role. Furthermore, we deal with flexibility needs by providing a dynamic architecture supporting runtime updates and transfer of roles and their rights. To this purpose, we leverage Non-Fungible Mutable Tokens (NMTs) for the representation of choreographies and business participants. NMTs are an advanced type of Non-Fungible Tokens (NFTs) designed to represent and manage digital assets dynamically whose state can evolve according to predefined rules. In this work, we specify choreographies and participants as NMTs, which are owned by organizations responsible for their respective duties and which can be updated at runtime. In this way, we decouple the initial specification of choreography from its execution, focusing on participants' certification. Indeed, NMTs provide digital proof of ownership, certifying each participant's role within the choreography. Furthermore, the same roles can be reused across different instances of the same or even entirely different choreographies, significantly enhancing the reproducibility and scalability of our proposal. To demonstrate the feasibility of the implemented architecture, we executed the various functionalities on an exemplificative canteen scenario, evaluating its performance. The rest of the paper is structured as follows. Section 2 introduces background concepts and Section 3 reports the state of the art. Section 4 provides an overview of the proposed architecture, its actors, components, and functionalities. Section 5 describes the implemented architecture while Section 6 evaluates its performance on the canteen case study. Finally, Section 7 concludes the work, touching up on future directions.

2. Background

Non-Fungible Mutable Tokens NMTs [14] represent an advanced extension of traditional Non-Fungible Tokens (NFTs), addressing challenges related to asset mutability, dynamic updates and changes regulated by policies. As defined by the ERC-721 standard [15], NFTs are designed primarily to track ownership of static digital assets. Each NFT comprises essentially two primary elements, *tokenId*, a unique reference to the digital asset and the *owner's identifier*, a blockchain address uniquely identifying the token holder. Furthermore, each NTF is associated, via the *tokenURI* parameter, with a cryptographic hash of the asset's representation or the asset descriptor, both typically stored through IPFS [1] as off-chain resources. The asset descriptor, commonly referred to as metadata [16], may include the cryptographic hash of the asset's representation and provides detailed information about the asset's characteristics.

NMTs overcome the limitations of hash commitment inherited by the NFT standard, enabling dynamic updates of the asset's descriptor stored on-chain throughout its lifecycle without compromising its cryptographic hash integrity. Specifically, an NMT comprises a series of smart contracts defining its main components associated with a pair of actors. An NMT (Figure 1) can be seen essentially as an NFT, which has the ability to generate a new asset, called *Mutable Asset*, with each minting operation. This

[1] https://ipfs.tech/

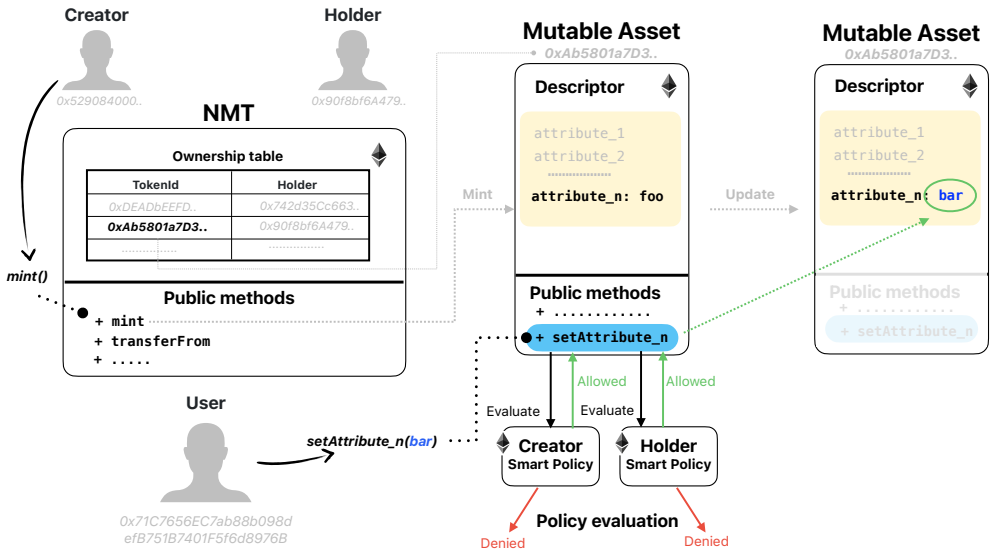

Figure 1: NMT overview.

asset represents a specific type defined by the NMT itself and is tracked along with its ownership within a logical table embedded in the smart contract. Also, the *Mutable Asset* is, in turn, a smart contract, from which its *tokenId* is derived based on its address. A *Mutable Asset* enables modifications through dedicated asset updater functions. The NMT manages the creation, ownership, and possible property transfer of the associated NMT-specific *Mutable Asset*. The pair of actors involved in this structure is: the *Creator*, who designs and defines the Mutable Asset and its NMT, and the *Holder*, who holds an instance of the Mutable Asset. The *Holder* can change over time following a *transferFrom* operation, as any NFT asset.

Access Control Systems (ACS) are designed to protect digital resources by ensuring only authorized users can access them. One of the most widely used models for managing access rights is Attribute-Based Access Control (ABAC). In ABAC, access decisions are based on attributes linked to:

- the *subject (S)* (e.g., their job role, department, or assigned projects).
- the *resource (R)* being accessed (e.g., file type or project association).
- the *environment (E)* (e.g., time of access or location).

For example, a policy might state: *"A subject S can access a file R only if it belongs to a project that has been assigned to S."*

To implement ABAC policies, many organizations use XACML (eXtensible Access Control Markup Language) [17], a standardized language that defines access rules in a structured way. XACML policies define *targets* that specify which requests the rule applies to. *Conditions* within the rules evaluate logical comparisons between attributes (for example, checking whether the user's department matches the file's owner). Each rule then enforces an effect, either permitting or denying access. Since real-world scenarios often involve overlapping rules, XACML includes conflict-resolution methods. For example, if one policy allows managers to edit a document but another restricts edits during an audit, the system uses predefined logic to determine which rule takes priority.

In NMTs, all the possible updates are securely regulated through an ABAC mechanism [18] which enforces dedicated policies, called *Smart Policies*, defined by the asset *Creator*, the original minter of the asset, and the current asset's *Holder*. These policies are implemented as smart contracts and govern the dynamic behavior of the asset, regulating how internal attributes can be modified. They are defined at two levels: the *Creator Smart Policy* and the *Holder Smart Policy*. The first, defined by the asset's creator, specifies the fundamental rules that determine which changes are allowed and under what conditions

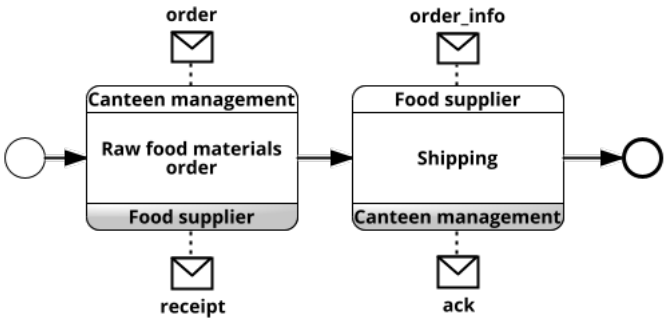

Figure 2: Public school canteen choreography.

they may occur. The second one is instead defined by the asset's holder which expresses additional preferences or constraints, as long as they remain consistent with the creator's original policy. Policies not only specify which changes are technically allowed but can also define who is authorized to make these changes and under which contextual or environmental conditions (e.g., external events, time constraints, or environmental factors). This layered policy model allows for a controlled but flexible approach to asset evolution, balancing authority between the creator and holder of the NMT. To define and express asset policies, NMTs rely on XACML, enabling Creators and Holders to work with familiar policy editor tools and avoid learning new languages. These XACML-based policies are automatically translated by the Smart Policy framework into executable smart contracts, following the methodology described in [19]. This process enables the dynamic generation of customized Access Control Systems, where each policy is translated into a dedicated contract that enforces the corresponding access control rules on-chain. Each Smart Policy has an `evaluate` method that takes three parameters as input: `subject`, `resource`, and `environment`. Based on these variables, the method assesses the conditions defined by the creator or the owner.

BPMN Choreographies BPMN choreographies represent how business participants interact in terms of the exchange of messages. These models provide a global view of the interaction coordination and can be seen as a sort of business contract. The most relevant elements used in choreography diagrams can be divided into control flow and communication. Typically, a choreography model is composed of different types of elements. Events represent the start and end of a choreography while sequence flows describe the control flow with gateways specifying alternative and/or parallel paths. Tasks define the interactions between participants and the related exchange of messages. They are represented as rectangles divided into three bands: the central one contains the task's name, while the others contain the initiator and recipient participants. Messages can be sent either by one party (One-Way tasks) or by both parties (Two-Way tasks).

To show an example of a choreography, we introduce the public school canteen scenario. The considered choreography is depicted in Figure 2 and represents the interactions for the supplying management of the canteen of a public school between the canteen management and a supplier. The process simply depicts the canteen sending the order of certain food raw materials to the supplier which responds with the receipt. At this point, the supplier sends the materials to the canteen which confirms the arrival. In this inter-organizational scenario, the involved actors require a reliable way to define and enforce their participation and duties within the procedure. As a public procedure, it is essential to maintain transparency, allowing all parties to demonstrate their participation and adherence to the process in a verifiable manner. Additionally, the procedure must be adaptable to changes, such as the replacement of a supplier due to the end of a contract or the inclusion of new messages due to a change in public regulation.

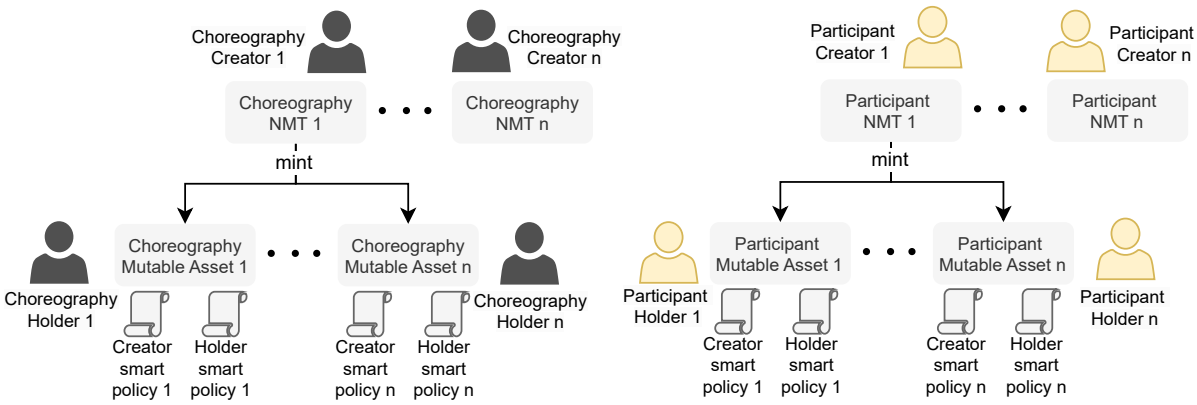

Figure 3: Actors and components of the ChorNMT architecture

3. State of the art

Blockchain has been widely used to support business processes, especially with the employment of smart contracts encoding and enforcing their execution [20, 21]. To create blockchain-based applications, the development has found model-driven engineering a prominent solution, relying on abstract models to reduce the complexity in the development and automate code generation [13]. Specifically, many works addressed the practical implementation of blockchain-based applications by means of the BPMN notation [13, 12]. Such solutions are implemented through smart contracts, the business logic and the exchange of information conceived as executable functions. These also enforce constraints about the sequence of actions to perform and the users in charge of them. During the process design, tools are typically used to manage participants and translate them into blockchain users, enabling them to interact only where they have responsibilities, thanks to role-based access control mechanisms. Over the years, these works have supported diverse approaches to participant and role management, also considering runtime changes [22]. The aforementioned works focus on the enactment of processes, including role management and support for runtime modifications. However, these solutions remain primarily focused on the execution phase, providing ad hoc solutions for participant management, mainly for access control.

In contrast, our work aims to decouple the design and definition phase of the choreography, including participants, their roles, and rights. Specifically, our approach does not cover the execution phase, which is out of the scope of this work. Instead, we aim to provide a dedicated solution for participant management. Our proposal includes an initial setup of the choreography and its roles, followed by runtime updates or role transfers. This is achieved through the use of NMT, which enables participants to hold ownership of NMT and assets associated with specific roles in various choreographies.

4. The ChorNMT Architecture

In this section we introduced the proposed architecture and its various elements to represent BPMN choreographies and participants.

4.1. Components

Figure 3 depicts the architecture, showing the key actors and components, which are divided into two entities: *choreographies* and *participants*. Each entity in the architecture, i.e., choreography and participant, is structured around two key smart contracts: an NMT smart contract, which defines the template, minting logic, and management operations; and a Mutable Asset smart contract, representing a specific instance derived from the NMT. These smart contracts are associated with the two specific

technical roles intrinsic to the NMT model itself: the *Creator*, who designs and deploys the NMT smart contract, and the *Holder*, who owns and manages a specific instance of the mutable asset.

Choregraphy entity - defines the general BPMN model that serves as a reusable blueprint for instantiating multiple instances of a specific process, each with its own lifecycle and set of participants. For example, the public school canteen procedure in Figure 2, managed at a regional level, represents a general choreography from which individual schools instantiate local versions tailored to their specific context.

In the proposed architecture, this general model is represented by a *Choreography NMT*, deployed on the blockchain and acting as a registry and controller for *Choreography Mutable Asset* ownership, in line with NFT but with mutable capabilities. This NMT supports the minting of *Choreography Mutable Assets*, each corresponding to a specific instance of the process (e.g., the canteen procedure in a school). The NMT structure enables evolution over time, including modifications to the BPMN model itself or to its associated BPMN roles, such as adding or removing participants to reflect organizational or regulatory changes.

Each *Choreography NMT* is created by a *Choreography Creator*, such as a central education agency in our example, who is responsible for defining the choreography logic and its governance policies. These policies, embedded in *Creator Smart Policy* and *Holder Smart Policy*, regulate asset-level changes. For instance, only designated school or central education agency offices might be authorized to update the local canteen procedure. Ownership of each *Choreography Mutable Asset* is assigned to a *Choreography Holder*, e.g., the Cyberville School, which is responsible for managing its local implementation.

This structure allows schools to customize their canteen procedures while maintaining a link to the general model. The mutability of the asset ensures that updates like introducing new supplier roles or adapting the model to nutritional policy changes can be applied without redeploying a new smart contract.

Participant entity - defines a generic participant type involved in a choreography and is used to represent specific actor roles within a choreography, such as a food supplier or a canteen management, as shown in Figure 2. In the proposed architecture, each *Participant NMT* acts as a reusable template for generating *Participant Mutable Asset* of the same type, for example, a food supplier, and each asset corresponds to a real-world organization fulfilling that BPMN role within a specific choreography instance.

The *Participant NMT* is defined by a *Participant Creator*, which could coincide with the entity responsible for the choreography. The creator specifies structural properties and access policies governing the evolution of the participant role. Once minted, *Participant Mutable Asset* are linked to one or more choreography instances (*Choreography Mutable Asset*s) and associated with real-world organizations acting as *Participant Holder*. For example, the food supplier role in the school canteen procedure may be represented by an asset held by "Supplier Company A", denoting its active assignment to that role. The asset includes the BPMN role definition, name, description, and other important details, and its related interactions.

Holding a *Participant Mutable Asset* represents the operational responsibility, and may be interpreted as evidence of an active engagement, even in the absence of an explicit legal contract. When the assignment ends or is reassigned (e.g., due to supplier turnover), the asset can be transferred to a new holder without altering the underlying choreography structure.

Participant Mutable Assets also support mutability: specific fields such as the supplier's name, service description or related interactions, can be proposed for update by any actor. However, a modification is applied only if both the proposed value and the identity of the requester satisfy the constraints defined by both the *Participant Creator* and *Participant Holder*. For instance, a company undergoing rebranding may adjust its metadata accordingly, provided the change is authorized by the Holder Smart Policy in accordance with the *Participant Creator*.

As with choreography assets, Participant Smart Policies (for both creator and holder) regulate lifecycle transitions and modifications, enabling secure and adaptable participant management across choreography instances.

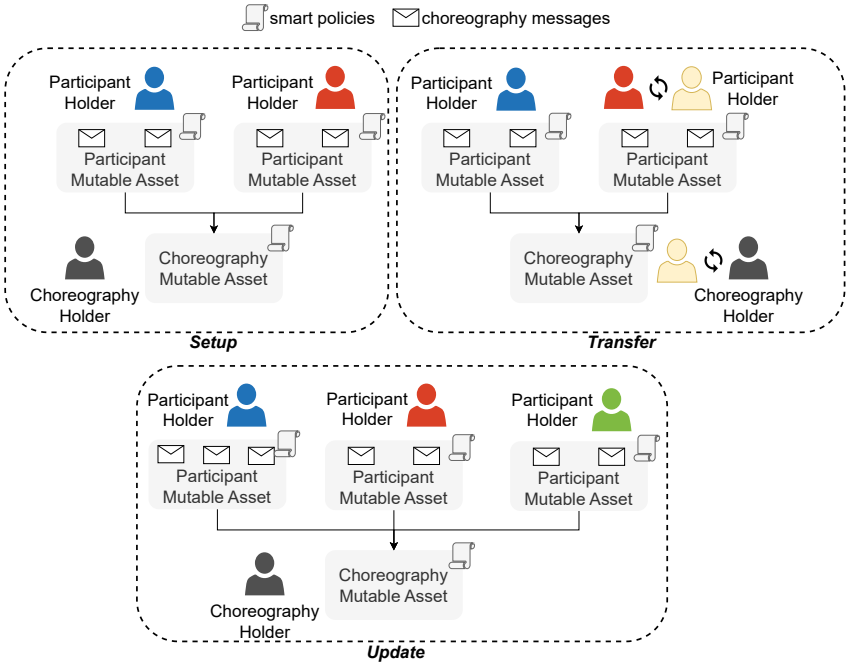

Figure 4: The ChorNMT functionalities

4.2. Functionalities

Here we outline the ChorNMT functionalities. In particular, we focus on the choreography and participant mutable assets, as they represent the core components of the architecture. Figure 4 shows the three main functionalities of the ChorNMT, respectively *set up*, *update*, and *transfer*.

Set up is the first functionality, where the Choreography and Participant Creators mint the corresponding Choreography and Participant NMTs according to the structure introduced in Section 4. As a result, the *Choreography Mutable Asset* and *Participant Mutable Asset* become available, representing the choreography instances and actual participants. In the public school canteen procedure, this functionality corresponds to the deployment and minting of the various NMTs and assets needed.

Transfer This functionality permits the transfer of ownership for the Choreography and Participant Mutable Assets. By inheriting NFT characteristics, the transfer enables the holders of assets to be changed. Considering the canteen procedure, at the participant level, the transfer can refer to the change in a supplier company due to the end of a contract. This leads to the transfer of the responsibility of the food supplier role, passing from Supplier Company A to Supplier Company B. At the choreography level, another transfer can refer to the management of the canteen procedure, passing from the Central Agency School to a municipality. To regulate such changes, the NFT approval mechanism specifies whenever the transfer of ownership can take place. If approved, the ownership transfer is authorized, and the corresponding operation is done on such an asset; otherwise, this possibility is denied.

Update the last functionality permits the update of the choreography at runtime, allowing for dealing with changes in the collaborative scenario. This is possible thanks to the mutable characteristic of the NMT. In this case, two different updates are possible. In the first one, a new *Participant Mutable Asset* can be included and linked to the *Choreography Mutable Asset*. In the second one, an already existing *Participant Mutable Asset* can be updated by adding or removing messages to exchange. In both cases, the ABAC mechanism is used to check whether the policies are satisfied, authorizing or not the operations. For example, an update could be about the inclusion of a new supplier participant for a specific raw material, leading to an update in the message exchange of the canteen management.

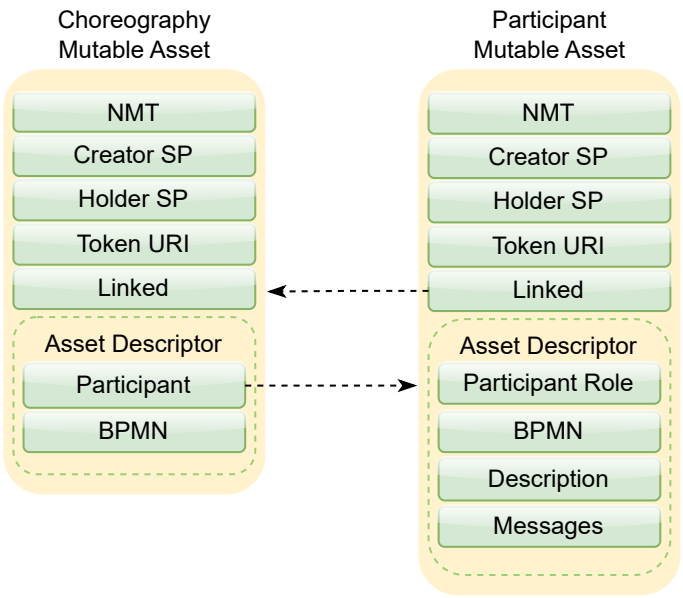

Figure 5: Structure of the *Choreography Mutable Asset* and *Participant Mutable Asset*.

5. NMT implementation

In this section, we describe the implementation details of the architecture we propose in 4 to represent choreography and its participants. The full code of smart contracts is accessible in the online repository[2]. The structure of NMTs and their assets is based on the implementation presented in [14], from which the base classes were reused and appropriately extended to include specific mutable attributes related to the mutable assets of the choreography and its participants.

Starting with the *Choreography NMT* and *Participant NMT*, these only contain general metadata and functionalities to mint related assets, keeping a reference to the different holders. For this reason, in this work, we focus on the *Choreography Mutable Asset* and the *Participant Mutable Asset*, whose structures are depicted in Fig. 5. In the following, we provide a general overview of the common structure, focusing on the *Asset Descriptor*, while all the other aspects can be found in detail in [14].

These assets share a core structure and main attributes:

- **NMT**: stores the address of the original NMT from which the asset is minted;
- **CreatorSmartPolicy**: stores the address of the smart contract containing the smart policy defined by the asset's creator;
- **HolderSmartPolicy** stores the address of the smart contract containing the smart policy defined by the asset's current holder;
- **TokenURI**: is the reference to the asset's off-chain metadata;
- **Linked**: when this asset is associated with another mutable asset, this variable stores the address of that asset's smart contract, which is linked to.

The *Asset Descriptor* contains specific updatable metadata for a particular asset, a specialization of a *Mutable Asset*, and can be customized for different NMTs and needs. For the choreography asset, standard content is represented by the following attributes, having corresponding set and get functions:

- **Participant**: is the list of the *Participant Mutable Assets*' tokenId (i.e., addresses), linking the choreography to the involved participants;
- **BPMN**: contains the XML representation of the BPMN choreography directly or through an external reference (e.g., IPFS link).

[2]https://github.com/d0na/BPMN-NMT-Roles

The *Participant Mutable Asset* is instead characterized by the following default attributes, having the corresponding set and get functions:

- **Name**: is the name of the participant role in the choreography;
- **BPMN**: contains the XML representation of the participant element in the BPMN model directly or through an external reference (e.g., IPFS link);
- **Messages**: is the list of messages that the participant is responsible for;
- **Description**: is the participant description, such as company name or additional data.

Listing 1 shows an excerpt of the *Choreography Mutable Asset*. The first element is the descriptor (Lines 1-4), which describes the content of the *Choreography Mutable Asset*. This includes the list of participants' addresses (Line 2) and the BPMN field (Line 3), which holds the IPFS URL of the BPMN model. The BPMN model is stored off-chain for convenience, allowing for efficient referencing without increasing the on-chain data size. Each attribute of the descriptor is provided with a dedicated *setter* method (Lines 6-9 and Lines 11-14) that protects its modification through the `evaluatedBySmartPolicies` modifier (Lines 6,11 and Listing 2). In case the policy is correctly evaluated, the setter function is allowed to update the associated attribute (Lines 7, 12) in the descriptor and emits an event (Lines 8, 13) to notify the state change within the mutable asset.

```
1  struct Descriptor {
2      address[] participants;
3      string bpmn;
4  }
5
6  function setParticipants(address[] memory _participants) public evaluatedBySmartPolicies(
       msg.sender, abi.encodeWithSignature("setParticipants(address[])", _participants),
       address(this)){
7      descriptor.participants = _participants;
8      emit StateChanged(descriptor);
9  }
10
11 function setBpmn(string memory _bpmn) public evaluatedBySmartPolicies(msg.sender, abi.
       encodeWithSignature("setBpmn(string,string)", _bpmn), address(this)){
12     descriptor.bpmn = _bpmn;
13     emit StateChanged(descriptor);
14 }
15
16 function getParticipants() public view returns (address[] memory) {
17     return descriptor.participants;
18 }
```

Listing 1: Excerpt showing the *Choreography Mutable Asset* Descriptor and some of its getter and setter

The conditions defined in the smart contracts regulate the asset changes, as determined by both the Creator and the Holder, and are initially expressed using XACML. These XACML-based policies are then translated into smart policies through a dedicated translation mechanism that follows a similar methodology to the one described in [18]. The resulting smart policies are deployed as fully functional smart contracts (*Smart Policy*), as also described in [14]. Listing 2 shows the `evaluatedBySmartPolicies` modifier, which acts as a protection mechanism for setter methods. It is based on the evaluation process of a given action, executed in this case by both the Creator's and the Holder's Smart Policy. The action is considered authorized only if both policies return a positive evaluation (Lines 20, 24). The *evaluation method* is the public entry point exposed by each Smart Policy to assess whether a given action can be executed. This method takes as input three parameters (Lines 19): the *resource*, representing the address of the actor invoking the action; the *action*, expressed in encoded form using `abi.encodeWithSignature` (e.g., `"setParticipants(address[])"`); and the *address* of the resource involved. Internally, the evaluation method verifies whether the action requested by the subject on the specified resource satisfies all the constraints and rules defined within the policy.

```
19 modifier evaluatedBySmartPolicies(address _subject, bytes memory _action, address _resource
       ) {
20     require(SmartPolicy(creatorSmartPolicy).evaluate( _subject, _action, _resource) == true
           , "Operation DENIED by CREATOR policy");
```

Actor	Action	Role
Central Agency School (CAS)	Designs the general choreography model for the school canteen procedure.	Choreography creator
	Initially owns the instance of the choreography mutable asset, acting as *Choreography Mutable Asset* holder.	
Cyberville School (CS)	Adopts the choreography model, receiving the ownership of the model instance after winning a public call. Acts as *Choreography Mutable Asset* holder after the transfer.	Canteen management
	Is the organization holding the *Participant Mutable Asset* of the corresponding role.	
	Manages the choreography instance because authorized by the CAS for updating the model.	
Supplier Company A Supplier Company B	External companies that provide meals or necessary materials for the canteen.	Food supplier
	Is the organization holding the *Participant Mutable Asset* corresponding to the role.	
	Food Supplier A is replaced with Food Supplier B, simulating supplier changes, transferring the *Participant Mutable Asset* ownership.	

Table 1
Actors of the public school canteen scenario and their actions.

```
21      if (holderSmartPolicy == address(0)) {
22          revert("Operation DENIED by HOLDER policy set to DENY_ALL");
23      } else {
24          require(SmartPolicy(holderSmartPolicy).evaluate(_subject, _action, _resource) ==
                true, "Operation DENIED by HOLDER policy");
25          _;
26      }
27  }
```

Listing 2: Modifier of *Mutable Asset* allowing Creator and Holder actor policies to be evaluated

6. Evaluation

To show the feasibility of the proposed architecture, we executed the implemented solution, evaluating its costs. To test the architecture, we simulated the adoption of the choreography model for managing a school canteen (described in Section 2), designed by the Central Agency School (CAS). The Table 1 lists all the actors and participants involved in the test, specifying their roles and the actions they perform.

In this context, the Cyberville School (CS), after winning a public call, is associated with an instance of a *Choreography Mutable Asset* via a dedicated *Choreography NMT*, handled by CAS. The ownership of this instance is then transferred from the CAS, the creator and previous holder, to the CS. From this moment, only the CS and the CAS have the authority to update the BPMN model of this instance, as stated by the Smart Policies. We tested the various functionalities of the system *setup*, ownership *transfer*, and *update* to evaluate the dynamics and costs of the architecture. In particular, we simulated the transfer in terms of participant Role change, such as the supplier company (e.g., replacing Food Supplier A with Food Supplier B), and in terms of administrative control over the choreography itself (e.g., transferring control from CAS to a CS). Updates of both *Choreography Mutable Asset* and *Participant Mutable Asset* were also tested by including a new dish supplier and modifying the BPMN process to adapt requests for dishes.

Table 2
Gas used analysis for canteen scenario.

Func.	Executed operation	AVG gas used	Total average gas used
Setup	Deploy SupplierNMT	5318860	28294841
	mint	2785951	
	SupplierMutableAsset		
	setMessages	125905	
	setDescriptor	86686	
	Deploy CanteenNMT	5319006	
	mint	2785951	
	CanteenMutableAsset		
	setDescriptor	86674	
	setMessages	125905	
	Deploy ChoreographyNMT	4775577	
	mint	2276921	
	ChoreographyMutableAsset		
	setParticipants	116893	
	setBpmn	89796	
	canteenManagement - CreatorSmartPolicy	914259	
	canteenManagement - HolderSmartPolicy	752221	
	choreography - CreatorSmartPolicy	542094	
	choreography - HolderSmartPolicy	525662	
	supplier - CreatorSmartPolicy	914259	
	supplier - HolderSmartPolicy	752221	
Transfer	***ChoreographyNMT***		196694
	transferFrom	95182	
	SupplierNMT		
	transferFrom	101512	
Update	Deploy ParticipantNMT	5318920	8495657
	mint	2785963	
	ParticipantMutableAsset		
	setMessages	125905	
	ChoreographyMutableAsset		
	setParticipants	84200	
	setBpmn	87623	
	SupplierMutableAsset		
	setMessages	93046	

6.1. Cost evaluation

The blockchain platform used to validate this implementation is a local Hardhat[3] Ethereum-based node with all smart contracts developed in Solidity. To perform the experimentation, we created one *Choreography NMT* and one *Participant NMT* representing, respectively, the choreography of the canteen procedure and the participant roles. These were then minted to obtain one *Choreography Mutable Asset* and three *Participant Mutable Asset* representing the instance of the canteen procedure for a particular school and the actual involved organizations. As previously mentioned, the choreography is intended to represent the collaborative process of canteen management, while the three participants correspond to the actors involved in this collaboration. As shown in Figure 2, two participants represent the Canteen Management and the Food Supplier, respectively, while the third is introduced to associate with a new task related to the provision of dishes.

Table 2 shows the experimentation results highlighting the executed operations in the various functionalities during the execution of the canteen scenario. Figure 6 depicts instead the obtained results. Analyzing the result, the setup is the first functionality and consists of deploying the choreography and

[3]https://hardhat.org/

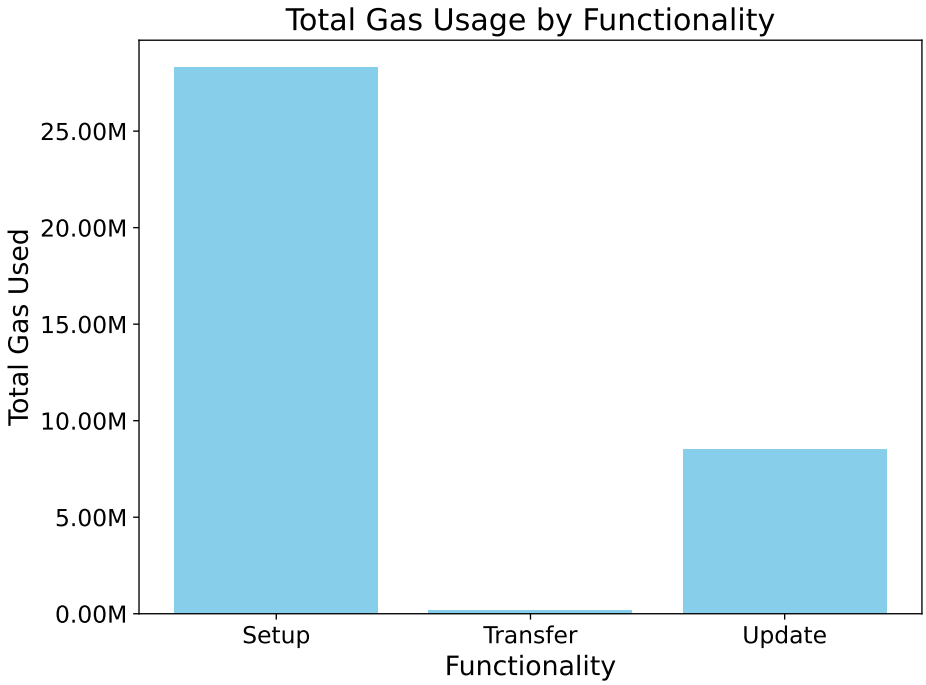

Figure 6: Gas used by executing functionalities on the canteen scenario.

participant NMTs, minting related mutable assets and setting the various data. This step consumed a total of 28294841 units of gas, with the deployments being the most expensive operations. Indeed, deployment and mining operations are certainly the most expensive operations, but are executed once and generate persistent artifacts. In fact, after being deployed, NMTs persist over time and can be minted multiple times to create different choreography instances. Other operations related to the initialization of asset data are instead cheap. Regarding the transfer functionality, it consumes a total of 196695 units of gas. In this case, the multiple transfers refer to the ownership change in both the Choreography and Participant Mutable Assets. As a result, this functionality is the simplest one and it does not show a relevant impact on the overall cost. Lastly, the update functionality consists of the minting for the new joining participant, its settings and the update in the already existing choreography and participants. This step does not include any new operation, but it foresees the execution of already existing ones, with the difference that those can be done at runtime. In the considered case study, the update consumed a total of 8495657 units of gas due to the execution of a mint operation.

In order to assess the economic sustainability of the proposed system, a cost analysis was performed considering the total gas consumption for contract deployment (setup), asset transfers, and subsequent updates. To translate these values into monetary costs, the average gas price recorded on the Polygon network in March 2025 was taken as a reference. According to publicly available data [23], the average gas price for that period was about 101 Gwei and approximately $0.23 per 1 POL. We can so derive the following estimated costs: $0.64405 for the setup functionality, $0.00045 for the transfer, and $0.193381 for the update. This analysis provides a quantitative basis for comparing the operational costs under the different interaction scenarios, highlighting the overall economic feasibility of the solution.

7. Conclusions and Future Works

Inter-organizational business processes involve multiple distributed participants collaborating toward shared goals. Certification and rights are crucial in this context, as they provide the foundation for accountability, traceability, and effective governance, especially when dealing with dynamic environments where roles and responsibilities may evolve over time. Blockchain has emerged as an enabling

technology, providing decentralization, trust, and auditability guarantees thanks to its transparency and immutability characteristics. In particular, BPMN choreographies have been used to describe the interactions among process participants, translated then into smart contracts. However, while existing blockchain-based approaches enforce inter-organizational process logic through smart contracts, they often overlook critical aspects such as role certification, rights, and runtime flexibility.

In this work, we proposed the ChorNMT architecture to certify choreography participants on the blockchain and manage their roles. We introduced the use of NMTs to represent both the choreography specification and the associated participants. NMTs provide verifiable digital ownership and can be updated dynamically, supporting runtime changes such as role transfers or attribute updates. By modeling both choreographies and participants as NMTs owned by responsible organizations, we provide a certified role attribution. Furthermore, roles and choreography components can be used across different process instances, supporting interoperability. To evaluate our proposal, we implemented the ChorNMT architecture and demonstrated its core functionalities using an exemplificative canteen scenario, evaluating its performance and showcasing the effectiveness of NMT-based choreography certification.

Future works relate to the implementation of the ChorNMT application supporting participants in the guided creation of NMTs and mutable assets. We also plan to extend the evaluation of the ChorNMT and perform scalability experiments to assess the performance under different scenarios.

Acknowledgment

This work was partially supported by project SERICS (PE00000014) under the MUR National Recovery and Resilience Plan funded by the European Union - NextGenerationEU

Declaration on Generative AI

During the preparation of this work, the author(s) used ChatGPT in order to: Grammar and spelling check, Paraphrase and reword. After using this service, the authors reviewed and edited the content as needed and take full responsibility for the publication's content.

References

[1] H. Bala, V. Venkatesh, Assimilation of interorganizational business process standards, Information Systems Research 18 (2007) 340 – 362.

[2] V. Venkatesh, H. Bala, Adoption and impacts of interorganizational business process standards: Role of partnering synergy, Inf. Syst. Res. 23 (2012) 1131–1157.

[3] K. Bouchbout, Z. Alimazighi, Inter-organizational business processes modelling framework, in: ADBIS 2011, Research Communications, Proceedings II of the 15th East-European Conference on Advances in Databases and Information Systems, volume 789 of *CEUR Workshop Proceedings*, CEUR-WS.org, 2011, pp. 45–54.

[4] J. Mendling, I. Weber, et al., Blockchains for business process management - challenges and opportunities, ACM Transactions on Management Information Systems 9 (2018) 1–16.

[5] C. D. Ciccio, G. Meroni, P. Plebani, Business process monitoring on blockchains: Potentials and challenges, in: Enterprise, Business-Process and Information Systems Modeling, volume 387 of *LNBIP*, Springer, 2020, pp. 36–51.

[6] C. D. Ciccio, G. Meroni, P. Plebani, On the adoption of blockchain for business process monitoring, Software and Systems Modeling 21 (2022) 915–937. doi:10.1007/S10270-021-00959-X.

[7] J. Vom Brocke, M. Rosemann, Handbook on business process management 2, Springer, 2010.

[8] M. Dumas, M. La Rosa, J. Mendling, H. A. Reijers, et al., Fundamentals of business process management, volume 1, Springer, 2013.

[9] K. B. Danilova, Process owners in business process management: a systematic literature review, Business Process Management Journal 25 (2019) 1377–1412.
[10] M. H. Larsen, R. Klischewski, Process ownership challenges in it-enabled transformation of interorganizational business processes, in: 37th Annual Hawaii International Conference on System Sciences, 2004. Proceedings of the, IEEE, 2004, pp. 11–pp.
[11] OMG, Business Process Model and Notation (BPMN V 2.0), 2011.
[12] F. Stiehle, I. Weber, Blockchain for business process enactment: a taxonomy and systematic literature review, in: International Conference on Business Process Management, Springer, 2022, pp. 5–20.
[13] S. Curty, F. Härer, H.-G. Fill, Design of blockchain-based applications using model-driven engineering and low-code/no-code platforms: a structured literature review, Software and Systems Modeling 22 (2023) 1857–1895.
[14] D. D. F. Maesa, F. Donini, P. Mori, L. Ricci, Protecting non fungible mutable tokens: an application in the metaverse, in: 2024 IEEE International Conference on Blockchain and Cryptocurrency (ICBC), 2024, pp. 443–451. doi:10.1109/ICBC59979.2024.10634340.
[15] ERC-721 - OpenZeppelin Docs, https://docs.openzeppelin.com/contracts/5.x/erc721, 2025. Accessed: 2025-03-11.
[16] Metadata Standards, https://docs.opensea.io/docs/metadata-standards, 2025. Accessed: 2025-03-11.
[17] OASIS, extensible access control markup language (XACML) version 3.0. (OASIS XACML TC,2013,1), https://docs.oasis-open.org/xacml/3.0/xacml-3.0-core-spec-os-en.pdf, 2013. [Online] Accessed: 2025-03-11.
[18] D. D. F. Maesa, P. Mori, L. Ricci, A blockchain based approach for the definition of auditable access control systems, Computers & Security 84 (2019) 93–119. URL: https://www.sciencedirect.com/science/article/pii/S0167404818309398. doi:https://doi.org/10.1016/j.cose.2019.03.016.
[19] D. Di Francesco Maesa, P. Mori, L. Ricci, Blockchain based access control services, in: 2018 IEEE International Conference on Internet of Things (iThings) and IEEE Green Computing and Communications (GreenCom) and IEEE Cyber, Physical and Social Computing (CPSCom) and IEEE Smart Data (SmartData), 2018, pp. 1379–1386. doi:10.1109/Cybermatics_2018.2018.00237.
[20] W. Viriyasitavat, L. Da Xu, G. Dhiman, Z. Bi, Blockchain-as-a-service for business process management: Survey and challenges, IEEE Transactions on Services Computing 16 (2022) 2299–2314.
[21] W. Viriyasitavat, L. Da Xu, D. Niyato, Z. Bi, D. Hoonsopon, Applications of blockchain in business processes: A comprehensive review, IEEE Access 10 (2022) 118900–118925.
[22] T. Lichtenstein, H. Atwi, M. Weske, C. Pautasso, Loose collaborations on the blockchain: Survey and challenges, in: International Conference on Business Process Management, Springer, 2023, pp. 21–35.
[23] Polygonscan, Ethereum gas price, https://polygonscan.com/charts, 2025. [Online] Accessed: 2025-03-11.