



POLITECNICO
MILANO 1863

SCUOLA DI INGEGNERIA INDUSTRIALE
E DELL'INFORMAZIONE

From business process to Corda R3: enforcing privacy and security of smart contracts

TESI DI LAUREA MAGISTRALE IN
COMPUTER SCIENCE AND ENGINEERING -
INGEGNERIA INFORMATICA

Author: **Alessandro Di Renzo**

Student ID: 975860

Advisor: Prof. Mattia Salnitri

Co-advisor: Prof. Giovanni Meroni

Academic Year: 2022-23

Abstract

Contracts among individuals and organizations regulate the rhythm of daily life. Some limitations related to contracts have been raised in terms of high costs, time consumption, and subjective interpretation issues. To address this concern, smart contracts have been introduced, i.e., automated computer programs that verify and execute contract terms based on defined conditions, being faster and more efficient compared to traditional contracts.

Since smart contracts define sequential activities to be executed, they can be seen as processes: for instance, a smart contract can regulate how a patient can book a medical appointment and the following actions of the healthcare provider in order to open an appropriate ticket. Due to the link between smart contracts and processes, smart contracts can be modeled as business processes with BPMN 2.0 standard.

Smart contracts' execution involves data exchanges, hence there is the necessity to include security properties related to the data treated: for instance, in a contract governing the conduct of a medical examination, documents containing the patient's medical results must be protected from the access of unauthorized entities.

Security properties linked to data confidentiality and enforceability of decisions are fundamental to be included in the contracts, since data and decisions have a central role in the execution of activities in a contract. Blockchain is one of the main technologies upon which smart contracts are executed. Based on the properties that characterize this technology, it can be used to enforce security properties.

In this thesis, we focused on the development of a method for the enforcement of security properties of data confidentiality and enforceability of decisions, by exploiting Corda, a blockchain that allows us to realize contracts based on the interactions among participants and executed in a private environment. The method consists of: (i) contracts are initially represented with SecBPMN2BC modeling language; (ii) mapping from BPMN Collaboration to BPMN Choreography and extension of choreography diagram to include security properties; (iii) mapping and transformation from choreography diagram into Corda contracts; (iv) validation phase through realistic cases.

Keywords: Smart Contracts, Blockchain, Business Processes, Corda R3, Security

Abstract in lingua italiana

I contratti tra individui e organizzazioni regolano il ritmo della vita quotidiana. Alcune limitazioni legate ai contratti sono state sollevate negli ultimi anni in termini di costi elevati, consumo di tempo e problemi di interpretazione soggettiva. Per risolvere questi problemi, sono stati introdotti gli smart contracts, ovvero programmi informatici automatizzati che verificano ed eseguono i termini del contratto in base a condizioni definite, risultando più veloci ed efficienti rispetto ai contratti tradizionali. Poiché gli smart contracts definiscono attività sequenziali da eseguire, essi possono essere visti come processi: ad esempio, uno smart contract può regolare il modo in cui un paziente può prenotare un appuntamento medico e le azioni successive dell'assistente sanitario per aprire un ticket specifico. A causa del legame tra smart contracts e processi, gli smart contracts possono essere modellati come i processi di business tramite lo standard BPMN 2.0.

L'esecuzione degli smart contracts comporta lo scambio di dati, dunque c'è la necessità di includere le proprietà di sicurezza relative ai dati trattati: in un contratto che regola lo svolgimento di una visita medica, i documenti contenenti i risultati medici del paziente devono essere protetti dall'accesso di entità non autorizzate.

Le proprietà di sicurezza legate alla riservatezza dei dati e all'esecutività delle decisioni sono fondamentali da includere nei contratti, poiché i dati e le decisioni prese hanno un ruolo centrale nell'esecuzione delle attività di un contratto.

La blockchain è una delle principali tecnologie sulla quale vengono eseguiti gli smart contracts. Sulla base delle proprietà che caratterizzano questa tecnologia, essa può essere utilizzata per far applicare e rispettare le proprietà di sicurezza.

In questa tesi, ci siamo concentrati sullo sviluppo di un metodo per l'applicazione di proprietà di sicurezza legate alla riservatezza dei dati e all'esecutività delle decisioni, sfruttando Corda R3, una blockchain che ci permette di realizzare una blockchain che permette di realizzare contratti basati sulle interazioni tra i partecipanti ed eseguirli in un ambiente privato. Il metodo consiste nelle seguenti fasi: (i) i contratti sono inizialmente rappresentati con il linguaggio di modellazione SecBPMN2BC; (ii) mappatura da BPMN Collaboration a BPMN Choreography ed estensione del choreography diagram per includ-

ere le proprietà di sicurezza; (iii) mappatura e trasformazione del choreography diagram in un contratto in ambiente Corda; (iv) fase di validazione attraverso casi realistici.

Parole chiave: Smart Contracts, Blockchain, Processi di business, Corda R3, Sicurezza

Contents

Abstract	i
Abstract in lingua italiana	iii
Contents	v
Introduction	1
1 State of the art	5
1.1 Processes representation and execution on blockchain	6
1.2 Secure data in blockchain environment	8
1.3 Enforcement of decisions	10
1.4 Assessment of secure smart contracts	12
2 Baseline	17
2.1 Running example	17
2.2 BPMN Collaboration	18
2.3 SecBPMN2BC Modeling language	21
2.4 BPMN Choreography	25
2.5 Corda	28
3 Method definition	33
3.1 Method description	33
3.2 Assumptions of method	34
3.3 Extended BPMN Choreography	36
4 From Collaboration to Choreography	43
4.1 Collaboration to Choreography transition	43
4.1.1 Implications	46
4.2 Application of mapping from Collaboration to Choreography	47

5	From Choreography to Corda	51
5.1	Conceptual mapping	51
5.2	Rules of transformation	54
6	Validation	65
6.1	Collaboration to Choreography	65
6.1.1	Realistic examples	66
6.1.2	Corner cases	75
6.1.3	Results	77
6.2	Choreography to Corda	79
6.2.1	Algorithms application	79
6.2.2	Comments	85
6.3	Discussion	85
7	Conclusions	87
	Bibliography	89
	List of Figures	95
	List of Tables	97
	List of Algorithms	99
	Listings	101

Introduction

In recent years, several studies have highlighted limitations related to contracts edited manually which are characterized by high costs, time-consuming and, frequently, contentious points subject to personal interpretation. On the contrary, there was the need to shift towards contracts released in a faster way with fewer intermediate stages. In order to design and automate the contract management process, Szabo in [34] introduced smart contracts as computer programs that verify the terms of contracts and execute them automatically based on predefined conditions. This innovation can eliminate the need for intermediaries in the conclusion of the agreement as the code itself acts as a regulator and allows cost saving with a faster and more efficient process.

Smart contracts define the conditions to be met in an agreement between two parties, also specifying the activities that the parties must respectively perform to fulfill the agreement. Due to this aspect, contracts can be seen as processes that specify the individual activities they are composed of and regulate their sequence: an example can be represented by a smart contract that regulates the actions a patient must take to book a medical appointment and subsequently the ticket opened by the healthcare provider. In this context, ensuring a certain level of security in the formulation of smart contracts assumes a primary role.

Security properties such as data confidentiality and enforceability of decisions have to be considered for reaching secure contracts. For instance, a contract that includes an analysis laboratory and a hospital structure in which results of tests relating to patients with serious illnesses are communicated: if those data were accessible by unauthorized entities, laws on the protection of sensitive data would be violated. Moreover, some decisions taken by specific entities need to be approved by other participants involved in the contract: a public administration tender is an example of a situation where competitors that compete to provide the best offer should have the opportunity to validate the final decision of the winning competitor.

Processes are represented with several graphical modeling languages: among them, one of the most popular is the BPMN 2.0 standard. Since contracts can be seen as processes,

the same modeling language can be used to model contracts. However, the standard does not incorporate security properties. Consequently, the standard has to be enriched with security properties proposed by Köpke et al. in [18], with the modeling language SecBPMN2BC which defines smart contracts as processes with security notations.

The issue is associated with the way to guarantee the enforcement of security properties in smart contracts, inheriting the requirements and specifications of these properties from the procedural definition of the contract.

One of the technologies on which smart contracts are implemented and executed is represented by the blockchain, i.e., a shared distributed ledger that records transactions in nodes linked to each other called blocks. Thanks to the properties upon which it is built, blockchain technology is able to ensure immutability and transparency to the information stored on the ledger. It can be used as a security mechanism for the enforcement of security properties: as stated by Zhang et al. in [43], blockchain is characterized by multiple attributes linked to security and it is necessary to adopt different types of techniques, each one with pros and cons, to provide a system that ensures a certain level of security.

In this scenario, the blockchain is able to replicate the context of business processes by ensuring a certain level of security regarding information exchanged by design: for these reasons, the choice has landed on the Corda blockchain [6], a distributed ledger developed by R3 that offers the possibility of realizing a private network and manage the process through a contract that regulates single steps of the execution based on the interactions among parties that characterized the process.

This thesis exploits Corda for the enforcement of two security properties related to smart contracts for modeling business processes: confidentiality of data and enforceability of decisions. The method proposed includes sequential phases:

1. contracts are initially represented with SecBPMN2BC;
2. mapping from BPMN Collaboration to BPMN Choreography [28] and extension of choreography diagram to represent security properties;
3. mapping and transformation from choreography diagram into Corda contracts following the conceptual model and the rules of translation conceived, while guaranteeing the enforcement of security aspects;
4. testing phase, where the effectiveness of the model proposed is evaluated through business process realistic cases.

The document is organized as follows: Chapter 1 focuses on the presentation of other works present that belong to the same field of research; Chapter 2 describes technical

instruments and baseline of work; Chapter 3 provides the description of the method proposed in order to enforce security properties in contracts through Corda blockchain; Chapter 4 details the first transformation between Collaboration and Choreography modeling languages; Chapter 5 is related to the adaptation of choreography diagram into Corda contract; Chapter 6 is related to the validation of the method in terms of correctness and scalability, highlighting its strengths and weaknesses; finally, in Chapter 7, we revise our work to outline possible improving areas.

1 | State of the art

This chapter presents research works in literature with the goal of guaranteeing the enforcement of security properties related to smart contracts through blockchain technology.

Blockchain technology brought a new concept of storing information on the network, without the necessity of intermediaries and, on the other hand, with the possibility of guaranteeing transparency, immutability, traceability and cryptographic security to the stored transactions. Since it acts as a distributed ledger shared among peers on the network, every node has the same knowledge of data and each update is visible to everyone: this is the case of public blockchains where users can join the network without any type of authorization and can participate to the validation of transactions. Two of the most known public blockchains are: Bitcoin introduced by Nakamoto in [26], the first blockchain available in the network, and Ethereum proposed by Buterin in [7]. They are opposite to private blockchains: in these networks, the access to the network is controlled and authorized by specific users and the identity of participants is known to others, such as Chain developed by Chain in [9]. Another category of blockchain is the permissioned one, which adds a layer of governance with respect to the public ones since there are users who can specify the roles of participants and the visibility of the data. Hyperledger Fabric managed by Hyperledger in [17] and Corda developed by R3 are the most common in this category.

The rapid spread of this technology started to enlarge fields of applications, in particular inter-organizational processes have been object of studies and works related to the possible automation of tasks and activities, however there was always a problem in the mutual trust: organizations want to be ensured about processes executed automatically that may involve sensible data or the exposition of internal methodologies, as stated by Mendling et al. in [25].

1.1. Processes representation and execution on blockchain

Recently, several researches focused on possible links between blockchain and business processes to demonstrate that different type of industries with related processes can be interested in automatize processes using smart contracts. As discussed by Yang et al. in [40], an example of real case of business process is represented by the construction field: in the work it is showed that adopting the public blockchain of Ethereum or the private one of Hyperledger Fabric, it is possible to overcome difficulties in money transaction or information exchange due to the loss of management control, helping the fragmentation and discontinuity in the process. In the paper is discussed how the blockchain technology can help to make efficient, traceable and transparent several parts of the project, avoiding the emergence of disagreements between clients and contractors. Moreover, advantages and disadvantages in the selection of two blockchains have been analyzed, highlighting that pros and cons are present in both choices and they are strictly related to the example taken in examination. This scenario confirmed by the study of Weber et al., in [39], where is discussed the problem of trusting in process execution using blockchain technology, both public and private. The solution was found using a set of components that monitor the business processes. The paper shows two different methods to simplify the collaborative process, correlated by main components such as interfaces or triggers that link the off-chain activities with the process executed on the blockchain. In these two studies, they showed different ways to exploit blockchain technology to execute processes, however, they don't refer to security properties with the technology itself which guarantees security aspects.

Another example of the industry is the finance one, as stated by Wang et al. in [38], where it is proposed loan on blockchain (LoC), a new method to face cyber attacks and ensure safety. This management system is based on the Hyperledger Fabric blockchain that uses smart contracts to guarantee more protection against web attacks. They tested the solution proposed in the Chinese poverty alleviation loan and created a model with centralized and decentralized ledgers, using locking and unlocking algorithms for smart contracts that allow them to execute the transaction automatically. Moreover, data privacy were made safe through digital signature and the presence of the oracles. In this case, the work was focused more on preventing web attacks to ensure the safety of loans than providing a mapping to represent processes in the blockchain environment.

As discussed by Azaria et al. in [2], blockchain technology can be used in the healthcare sector to overcome the inefficiency of the bureaucratic system. The paper shows the creation of a new decentralized management record system, MedRec, to store and handle

electronic medical records (EMRs). Based on blockchain properties, the system, composed of three layers of smart contracts, records medical information of patients on different nodes of the blockchain. By applying a mechanism to retrieve pieces of information related to the same patient, the system is able to reconstruct the medical history of a defined patient guaranteeing, in the meanwhile, a high level of safety on the information stored. This is an interesting application of blockchain technology, even if there is no reference to map processes due to the different purposes.

To facilitate the move from process to smart contracts, as illustrated by Berry and Milošević in [4], business contract constraints can be transformed into expressions in a choreography standard. With an example of cross-organizational process the contract terms and boundaries are converted into choreography notations that control the process to guarantee compliance. This work addresses part of the problem that involves the use of choreography modeling language, even if our work goes beyond the representation in the choreography standard.

Instead, in work made by García-Bañuelos et al. in [13], it is described as a method that creates an efficient method for executing business processes defined in the standard Business Process Model and Notation (BPMN). In particular, they present a method for translating BPMN processes into Solidity language (the programming language supported by the Ethereum blockchain to generate smart contracts). To obtain the transformation there is a halfway step represented by Petri Nets that performs all the checks to verify the accuracy of the process before the definition of smart contracts with the blockchain technology. The study proposed the use of blockchain in order to execute business processes using Solidity as a programming language to code smart contracts: the difference in comparison with our work is that we added to this process the enforcement of security properties which represents a further step.

In order to map collaborative processes in blockchain environments, new model-driven approaches have been presented and tested on the Ethereum blockchain, such as Lorikeet, which takes as input choreography diagrams and generates smart contracts; or again, Caterpillar, an approach that is able to deploy a collaborative process on-chain, as stated by Di Ciccio et al. in [12]. This work refers to the mapping part from the choreography diagram to the blockchain, specifying that the blockchain used in this work is public, while in our work is permissioned, and that the security of contracts generated is not deepened.

Always in a collaborative environment, Loukil et al. in [23] provided a different method for translating processes into smart contracts, using CoBuP, a decentralized Collabora-

tive Business Process execution architecture that provides results following blockchain properties. It is composed of two separate phases as follows: initially, an interpreter of BPMN generates a generic smart contract that creates a process instance and, in a second moment, it is updated dynamically with process specifications. This work experiments with a new methodology for translating business processes into contracts, starting from generic contracts and then enriching them with details relating to the process itself: in this type of approach, which favors adaptability and reusability, there may be the problem of failing then to represent all the constraints starting from a generic contract. For this reason, in our work, we preferred to recreate a situation as close as possible to the starting one in each mapping, trying not to lose information in the transaction processes.

1.2. Secure data in blockchain environment

While tangible advantages have been reported in the use of blockchain and smart contracts, the topic related to security of data and the need to enforce security properties remain open arguments upon which various researchers focused on.

The need to deal with security properties when business processes are mapped and executed on blockchain, private or public, is explained by the work of Carminati et al. in [8], with the use of blockchain technology in support of secure inter-organizational business processes, to overcome the actual methods attempt to guarantee security in the execution of processes. With the help of blockchain technology, they were able to keep track of every exchange between participants of a network, called transactions, that form the so-called chain of blocks validated by network participants through Proof of Work (PoW). This work highlights the possibility of managing the process with the execution in a blockchain environment, but they didn't define a proper method for evaluating specific security properties in business processes.

As stated by Lin et al. in [20], the coming of the Industry 4.0 era requires some adjustments, so they proposed a new method for secure mutual authentication based on blockchain features, called BSeln. The work shows the high level of security and privacy obtained with the proposed system, with enforcement of access policy characterized by low costs of operations. This work focuses on the enforcement of a security property like access control, even if they are referring to other types of properties with respect to our work.

From these studies, it is clear the necessity to define security properties when business processes are defined through standards such as BPMN or UML. Several works proposed a method to extend BPMN notations to include aspects related to security: in the work

of Altuhhov et al. in [1], it has been highlighted the importance of security properties in business processes environment by proposing a method to extend BPMN to better include concepts of assets, risks and threats in order to develop security requirements to secure important assets. This work focuses on extending BPMN in order to manage risks and threats linked to assets in a process: this can be compared to the first part of our work with the definition and inclusion of security properties but we added even a practical way to enforce them through the blockchain.

Another work that follows the same direction of enriching BPMN standard with security requirements was made by Rodriguez et al. in [33], where it is provided a metamodel to represent graphically security aspects in business processes. They didn't focus on the actual execution of processes, but they tried to incorporate security requirements in BPMN standard enlarging the range of expressivity of the standard and allowing business analysts to add their own security requirements. In this case, what is missing is the definition of a precise set of security properties to represent in BPMN, because allowing personal information may mean changing the requirements multiple times.

On the other hand, the approach discussed by Zareen et al. in [41], focused on defining all threats that an IT system can face, from insecure network services to data input injections, and proposes a framework to represent business processes enriching the BPMN notation with threats detected. The framework is then tested in the manufacturing industry, showing that it can cover a great number of security worries in processes inside organizations. This work has focused on detecting all possible threats related to the execution of a business process, but their work has been limited to ascertaining that a possible threat is present and not providing a secure execution environment as proposed in our work.

Considering works that started from a defined standard and try to include security aspects, as illustrated by Vivas et al. in [37], it is proposed a UML-based business process-driven framework for the development of security-critical systems that shows possible threats related to the trade-off between security and functionality when defining security requirements for a system. A similar approach is suggested by the work of Lodderstedt et al. in [22] is aimed to define a modeling language for the development of secure systems with UML through role-based access control. It describes how to use UML to specify security requirements in modern systems that are well-suited only for static design models. This system language is based on an extended model for role-based access control (RBAC), but to overcome lacks of methodology they introduced the authorization constraints, a precondition for granting access to an operation, they defined such limit using the Object Constraint Language (OCL). SecureUML is a combination of the main features of

these two systems. As visible in these works, security requirements are defined also using UML as a standard, which is different from the standard that we have used in our work, i.e., BPMN, but it provides us with a methodology with which to enrich already existing standards with additional requirements.

Always focused on the extension of UML2.0 to include security requirements from a business analysts' perspective, is the work presented by Rodríguez et al. in [32]. In this study, they applied the extended UML2.0 to a typical healthcare business process to show how business process modeling is the key to carry on and enhance the way business processes are executed. The step made by using UML 2.0 is to be sought by using activity diagrams which improve the entire business process representation. Their approach is based on Model Driven Architecture (MDA), defining early requirements identification that allows to carry out independent specifications of the implementation. Since UML is not the modeling language chosen for the representation of processes in our work, these works show that the security aspects are included in different standards with different methodologies of extension.

Other research drew attention to privacy requirements regarding data accessed on business processes, as discussed by Labda et al. in [19]. The concept of privacy is split into four fundamental aspects such as data, user, action and purpose. In the method proposed there is a distinction between users that should or shouldn't access data during the evolution of business processes based on actions and purpose of data. This is a work that can be examined for the management of data in business processes, with the list of participants that can access or not specific data and when they are going to access it.

As stated by Pullonen et al. in [30], from a model called PE-BPMN (Privacy-Enhanced BPMN) which analyzes business processes and discover critical points for privacy of data in the process, they provided a method to detect and solve possible cases of privacy leakage with technologies like mobile applications where private data leakages are performed. System builders were assisted to make better decisions on the privacy solutions that they proposed at the beginning stages of development, letting auditors check existing systems. The framework proposed can be a valuable tool in order to detect our BPMN and find if there are some vulnerable points.

1.3. Enforcement of decisions

Decisions in a process modify the successive actions for entities that suffer or take them and their relevance in business processes mapped into blockchain context has been analyzed and discussed in several works as [14] by Haarman et al., where it is proposed a method to

automatically represent and verify the correctness of decisions taken in processes executed on Ethereum blockchain to ensure that decisions taken are more secure, more transparent, and better auditable. The method starts from the representation of decisions in a process using the Decision Model and Notation (DMN) standard by OMG [29], then it is translated into smart contracts with a mapping into Solidity language and Ethereum blockchain. The main problem in this approach is related to the use of public blockchain to save data expressed in the contract, which, in many situations, have to be accessed only by certain participants, they can't be worldwide.

Analyzing the work proposed by Nikaj et al. in [27], it has been proposed a method to enforce decisions deriving from exclusive gateways in BPMN choreography diagram to ensure that every participant affected by the decision has the same understanding of data. This is achieved through a RESTful decision service, which implements a REST interface that interacts with every participant in the choreography tasks affected by a decision. The approach allows to separate process and decision logic and represents a comparison for the enforcement of decisions property, while, on the other hand, they don't cover the aspect related to smart contracts.

The enforcement of decisions is discussed even in the work conducted by Haarman et al. in [15], where the approach detailed is aimed at the enforcement of decisions in a collaborative process executed by a smart contract on Ethereum blockchain. The approach is composed of two phases: the operation phase, where the decision is executed locally with an agreement of participants regarding input and data consumed to take the decision; the conflict resolution phase, where the agreement is not reached on the output of the decision and the conflict is solved by the smart contract itself at cost of revealing the logic under that decision. The purpose was related to the overcoming of limitations of exposing data of decisions in public blockchains with a condition, i.e., the conflict on the output, upon which the confidentiality of data is violated. Referring to our work, this kind of solution proposed is not applied, since the enforcement of properties related to decisions and data have to be preserved in all cases.

Taking decisions often means examining sensible or private data and then, based on results of performed analysis, selecting a path rather than another one. In collaborative processes, entities pretend correct behaviors from counterparties especially when decisions are taken. Many situations that involve decisions require that decision logic and relative data can't be shared with all entities, on the contrary they have to be protected by external accesses and the type of blockchain, public or private, can affect the exposition of data related to decisions: as reported in [15], it is needed to add a certain level of confidentiality on the data upon which decisions are taken and this can be made by splitting the decision into

more phases for the sharing of the logic model and the verification of the correctness of decision itself.

Other researchers worked on the separation of decision from the business logic, as discussed by Zarghami et al. in [42], with the proposal of a decision service that is able to react quickly to changes with a real-time adaptation to provide dynamic updates to the process. They proposed a method that utilizes the decision service to support both synchronous request-response and asynchronous interactions in order to design an adaptive service provisioning architecture and decision service template for different application domains. Limitations of this work can be found at the computational level, with a high request of resources and, related to our work, it can be analyzed to understand how decisions can impact at different business levels, but it doesn't solve our problem linked to security property and the enforcement of such properties. Other works concentrated on analyzing DMN and the coexistence with BPMN, with a possible mapping between these two standards, proposing different techniques to deal with decisions in processes: in [3] by Batoulis et al., it is presented a semi-automating model to identify decisions logic in process models, starting from BPMN, and derive DMN model with the successive implementation. They didn't consider AND gateways since they don't directly influence decisions. As central part of the work, they focused on extracting business logic from BPMN and, then, mapping it with DMN with a direct representation of the logic behind the task. They tested their model in 956 real-world cases with a positive result. This type of work can provide an approach in terms of how to model decisions, but they are not related to security properties and the application in the blockchain context.

1.4. Assessment of secure smart contracts

A related field of studies is constituted by assessing the correctness of the smart contracts generated. For instance, as reported by Delmolino et al. in [11], in order to avoid common mistakes and realize secure smart contracts, developed for the Ethereum blockchain, from the point of view of programming, they documented several guidelines that can be helpful to check if the contract has been programmed correctly. This kind of work can be related to the final part of our work, where we check if there are some criticisms in the generation of smart contracts, but we have to change the context from Ethereum to Corda to refer to our context and try to match the properties discussed.

Instead, as stated by Tsankov et al. in [36], in order to evaluate smart contract behaviors performed during the execution of code and in order to avoid unpredicted situations, they proposed a security analyzer, called SECURIFY. It is a scalable and fully automated

security analyzer designed for Ethereum smart contracts that is able to determine whether a contract behaves safely or unsafely through two main steps: at the beginning, it is conducted a symbolic analysis of the contract's dependency graph to extract semantic information of the code and then it is checked the compliance and validation pattern to establish if a given property is valid or not. It has been evaluated with real cases of smart contracts and SECURIFY highlighted its capacities to verify the correctness of smart contracts and identify critical violations. This method proposed can be used as a final assessment in order to determine if our code presents critical patterns or possible mistakes, even if it is not referred to our blockchain Corda.

An original method based on Finite State Machine (FSM) approach has been discussed by Mavridou and Laszka in [24] to ensure the generation of secure smart contracts starting from the initial phase: they developed a graphical editor that allows contract design using FSMs and then they generate automatically the code through a code generator. In addition, they created a collection of plugins that developers can incorporate into their contracts focused on security measures to prevent common vulnerabilities such as reentrancy and unpredictable states. This work offers support in all phases of smart contracts generation, however, they don't ensure the enforcement of specific security properties as we propose in our work, thus the tool couldn't address our specific issues in contract design.

In order to avoid possible financial losses caused by the deployment of insecure smart contracts, Zupan et al. in [44] introduced a comprehensive approach to design, develop, and verify secure smart contracts incorporating a modeling tool-set based on Petri Nets. The framework comprises four main components: a visual modeling engine, an execution layer for simulating Petri Nets workflow and executing transitions, a verification and validation engine to check specific properties such as deadlock and workflow soundness properties, and a translation engine for generating smart contracts from the modeled Petri Nets workflows. They performed an evaluation of the approach using a specific supply chain use case to simulate and test the workflow or business requirements providing the confidence that the smart contract is modeled as required and then it is translated into lines of code with Solidity language to be deployed in Ethereum blockchain. This work provides another opportunity to generate secure smart contracts using Petri Nets, but there are some limitations in the approach that can lead to possible mistakes in the execution phase: for instance, the code generated is not optimized and the template generated for the smart contract represents only a starting point that has to be enriched by external developers analyzing the business logic behind the contract.

Another work that provides a method for the assessment of smart contracts by exploiting

properties of Coloured Petri Nets (CPN) is the one discussed by Zhentian and Jing in [21]: in this case, the method proposed not only analyzes the static logical structure of the contract but also simulates the dynamic interaction of the user's malicious behavior, which can verify whether the contract has a vulnerability; moreover, the tool provided can monitor the status of each step of the contract execution, making it easier to discover potential vulnerabilities in smart contracts. Since this work refers to an analysis after the generation of the smart contract can be only taken into consideration for the simulation phase after having generated the smart contract, not in previous phases.

Petri Nets are also used in the process of translation from BPMN to Solidity language by Wen et al. in [16] with the purpose of reducing the compensation gas linked to transactions by producing an optimized smart contracts template starting from BPMN processes and by adapting it to specifications of Petri Nets before generating the smart contract code. Initially, the method proposes to extend BPMN business process model to Petri Nets and then simplify it by identifying fusion tasks within the BPMN model and streamlining the corresponding fusion nodes while ensuring the integrity of the original business process. The experimental results demonstrate the effectiveness of the method in reducing gas consumption by an average of 15% for blockchain-based business process smart contracts. This work can be analyzed for comparison regarding the translation of BPMN to Petri Nets and then into smart contract code, even aiming at a different purpose such as reducing consumption of gas with respect to our work.

To assess the correctness of smart contracts exploiting other methodologies, the work presented by Bhargavan et al. in [5] can be explored: they presented a framework for analyzing and verifying the runtime safety and functional correctness of Ethereum contracts through F^* , a functional programming language designed for program verification. The experiments indicate that the tool is sufficiently flexible to capture and prove properties relevant to contract programmers. Our approach, employing shallow embeddings and type-checking within an existing verification framework, is convenient for exploring the formal verification of contracts written in Solidity. However, the missing part in the paper is related to security properties that have to be discussed before coding, in contrast, this paper is focused on providing correctness only on the programming code side.

The automatic generation of smart contracts deployed in Hyperledger Fabric blockchain is discussed by Takaaki et al. in [35], where it is presented an approach based on a Controlled Natural Language (CNL) that provides a formal model used to generate smart contracts: the article proposes an automated technique to generate smart contracts from human-understandable contract documents using controlled natural language (CNL) and document templates to create a formal model representing contract terms, conditions,

and procedures. The formal model is then translated into executable smart contracts for Hyperledger Fabric. The approach was evaluated through real-world case studies of different contract types, demonstrating its feasibility and effectiveness. This work refers to the generation of smart contracts in the Hyperledger blockchain, thus the permissionless characteristic is the same with respect to our context, however, it doesn't examine security properties in the generation of smart contracts.

As stated by Choudhury et al. in [10], researches focused on an automatic smart contract template generation framework that uses the structure of Abstract Syntax Trees (AST) to include required constraints into the template. Their approach involved automatically generating smart contracts from domain-specific knowledge bases, using ontologies and semantic rules that were tailored to the specific domain constraints. The description provided by the ontology and rules were used to manipulate the AST of the template to align it with the specific requirements. They tested the approach in two real cases: a clinical trial protocol and a car rental criteria, achieving in both cases the generation of contracts and the enforcement of the defined rules. As can be seen from the document, the approach proposed was at the beginning of development and required further research to be enhanced, as they needed the presence of an expert figure to complete the final stage. Again, in the generation of smart contracts the part related to security properties has not been mentioned, so it can't be adopted into our context.

2 | Baseline

In this chapter they are presented all technical concepts and notions necessary to the comprehension of work discussed in Chapters 3, 4 and 5.

Section 2.1 describes a realistic example that will be taken as a reference point to showcase the transformation stages from the initial definition to the generation of the contract in Corda. Section 2.2 introduces of the collaboration modeling language enriched with an application of language applied to the contract presented in Section 2.1. In Section 2.3, it is introduced the language for the graphical notation of security properties adopted in the definition of the model. Section 2.4 presents an overview of the choreography model representation with the description of single elements of the choreography diagram. To end up, Section 2.5 presents the main concepts related to Corda and the structure of contracts realizable with this technology.

2.1. Running example

Contracts can be represented as processes due to their characteristic of representing specific activities to be carried out in a certain order, with a certain sequence that involves multiple participants that have to coordinate their operations in order to advance in the process. In order to define a running example, we are going to represent a contract that is composed of different steps in order to achieve a certain situation known to all participants. The scenario chosen to represent this example is composed of a Company Employee, an Health Care Fund and a Medical Office. The contract describes the request, successful or not, of an affiliated visit by the Company Employee to the Health Care Fund that is going to contact the Medical Office associated with that specific pathology. The collaboration diagram of the running example is described in Section 2.2.

Initially, it defines what are the activities that the Company Employee has to perform in order to send a well-structured request of an affiliated visit to be evaluated by Health Care Fund. Then, it describes what happens on the Health Care Fund side before sending a positive or negative response to the employee who submitted the request with the doc-

uments. In this initial phase, the first interaction has been regulated and the constraints are added to the operations carried out by both participants in order to coordinate the flow.

Furthermore, it is specified that a dual scenario can occur based on the decision made by Health Care Fund after analyzing the documentation: the request can be accepted, and in that case, certain operations will be performed, or it can be rejected because it is deemed unsuitable, leading to a different direction. Therefore, it is necessary to regulate both possibilities and ensure that the operations defined in one branch do not occur in case the other branch is chosen.

Especially in case of request accepted, another participant, Medical Office, intervenes in the sequence of operations by communicating with both other participants: first, by obtaining patient data from Health Care Fund, and then by sending available booking dates to the Company Employee. In the described scenario, defining constraints for data access and the sequence in which activities must be carried out is of primary importance. In this way, the definition of contract has to take into consideration the properties that activities must ensure and not allow situations not described in the contract.

2.2. BPMN Collaboration

BPMN Collaboration is a modeling language that aims to represent participants and relative tasks performed in a business process. Participants are represented by pools, generally one for each participant. A pool may contain more lanes to highlight two different participants of the same organization. Since pools act as containers for the activities of business processes related to that participant, it is important to specify that a pool can show the activities performed in a detailed way or act like a "black box" by hiding them. In our work, we refer to collaboration diagrams that show detailed and sequenced activities and tasks executed by participants, from which we are going to perform successive transformations in order to generate the contract in Corda.

In this sense, we are going to describe the main elements observable in a process, starting from the concept of event, represented by a circle: based on when they affect the flow of activities, it can be a start, intermediate or end event; on the other hand, based on the type of the event to describe, it contains a symbol representing, for instance, a message, a timer, a signal or a condition which triggers other activities.

Regarding the activities, represented by rectangles, they refer to operations or in general "works" performed by a participant in the business process. They can be atomic or not

and where they are atomic they are called tasks, which is the lowest possible level of detail used to describe an activity.

Gateways are represented by a diamond shape within an icon inside: they are used to indicate possible divergences or convergences of different paths in a flow. The internal icon describes the type of gateways we can exploit to manage more scenarios. The first type of gateway is the exclusive one, which is depicted with a greek cross and indicates that the possible paths are mutually exclusive, thus, based on the condition, only one path among those available will be taken. The same behavior can be seen in the event-based gateway, indicated by a circle within a pentagon, where only one path is selected among those available, but the choice of the path is based on a particular event. Other types of gateways are inclusive parallel ones: the former is depicted with a circle within the diamond and it indicates that one or multiple paths can be taken, always based on the condition; the latter, instead, is represented by a classic cross and it means that all possible paths will be taken simultaneously. The last type of gateway is the complex one, depicted by an asterisk, which means that to handle that particular type of situation in a business process, they are needed more than one "simple" gateway.

The sequence flow is represented by an arrow and it indicates the sequentiality of the activities in the process and it provides a precise order to the process. It links two different activities and, based on the nature of the activities from which it comes, there are different types of sequence flow such as uncontrolled, conditional and default flow. The message flow shows the flow of messages between two different participants, thus two different pools.

Data objects are depicted by a sheet with a bent corner and they refer to data that are required or produced as output by a certain activity. Furthermore, they can represent a singular or a collection of objects, based on the scenario they are introduced.

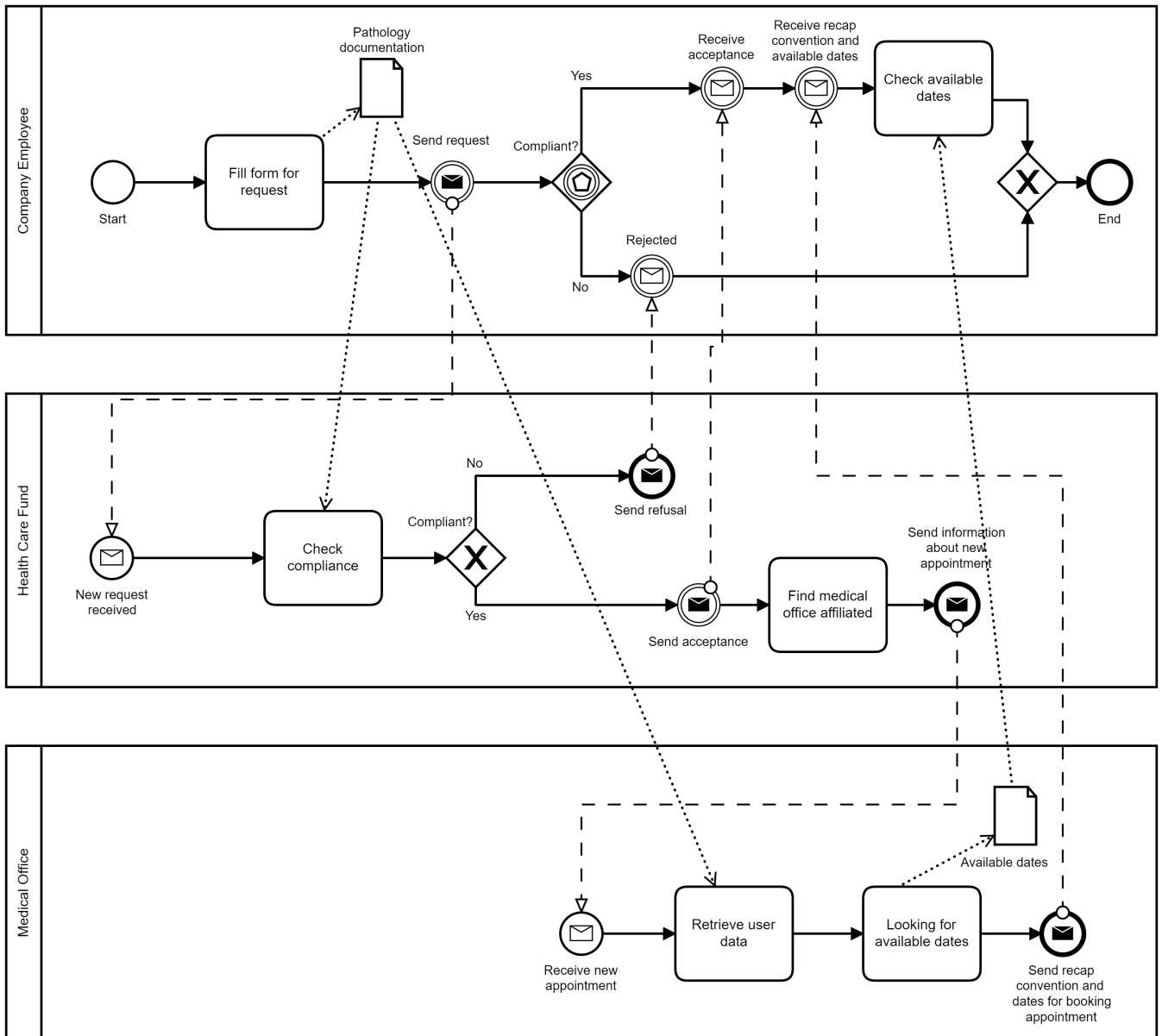


Figure 2.1: Affiliated medical visit

Going into detail about what happens in the contract described in Figure 2.1, the agreement is defined among three different participants: Company Employee, Health Care Fund and Medical Office. The contract represents the possibility for an employee to require a medical visit through the health care fund by taking advantage of the agreement reserved for a certain category of workers.

Initially, the employee fills out the form with their personal information and attaches a file containing documentation of their medical condition to verify if it falls within the list of conditions for which a subsidized visit can be requested. Subsequently, this form

with the attached file is analyzed by the fund, and the decision is communicated to the employee. In case of a negative outcome, the process concludes with the communication of the refusal. In case of a positive outcome, the fund communicates the acceptance of the visit and takes charge, contacting the affiliated medical office to request a new appointment and providing the employee’s documentation. Finally, the medical office examines employee data and sends her a file with a recap of the convention and available dates for scheduling the visit.

2.3. SecBPMN2BC Modeling language

Our work is based on the enforcement of security properties related to data confidentiality and the enforceability of decisions. These properties are specified in the model-driven approach proposed by Köpke et al. in [18]: it is an extension of BPMN 2.0 standard including security notations towards secure smart contracts to be deployed in a blockchain environment.

Smart contracts are represented using a graphical modeling language that allows to represent contracts as business processes with security requirements. It defines a workflow to design and deploy secure business processes expressed in Figure 2.2.

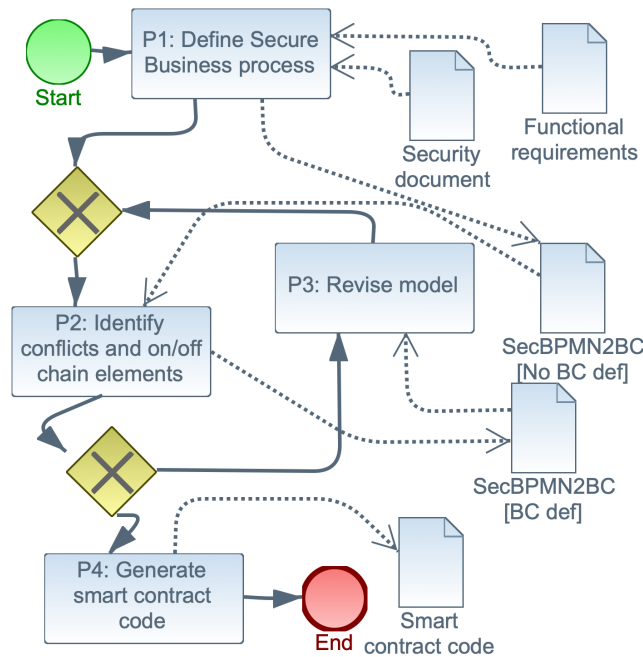


Figure 2.2: Secure business processes definition workflow from [18]

Among security properties discussed in the approach, our enforcement work is based on:

confidentiality of data, i.e. privacy of data objects, and enforceability of decisions.

In order to restrict read access to data objects and messages that are stored on-chain there would be necessary some supplementary methods such as encryption or on-chain storage channels. However, implementing these measures may restrict the blockchain system's capacity to validate transactions based on data. For these reasons, modelers should have the ability to explicitly specify read-access constraints for data objects and messages.

Privacy spheres related to data objects and messages have been specified to model read access requirements following a restrictive-based scale. A participant is in the strong-dynamic sphere of a data object, the most restrictive sphere, if that participant will execute a task reading the data value written by others. Instead, a participant is in the weak-dynamic sphere of a data object, if that participant can execute a task reading the data value written by others. More relaxed spheres are static, private and global: a participant is in the static sphere of a data object if that participant executes any activity in a business process accessing the data object, while it is in the private sphere of a data object if it is a participant to the process in which the data object is present. The global sphere refers potentially to all participants in the world, but it is designed for public blockchains, thus it is not applicable in the context of Corda because it would mean opening the network to external participants, which is against the principles of Corda. Privacy spheres are listed in Table 2.1.






Symbol	Level of privacy
	Public
	Static
	Private
	Strong dynamic
	Weak dynamic

Table 2.1: Privacy spheres

The approach aims to pursue proactive online enforceability, to make non-contractual behavior impractical. However, verifying only the feasibility of prescribed execution traces does not ensure faithful decision-making. Since the number of participants that are required to validate decisions depends on the single case, there is needed a security property that includes the set of participants with the duty to validate each decision. The requirement for enforceability of decisions is represented by a train symbol accompanied by an additional circle that specifies the desired level of enforcement. This annotation can be attached to conditional exclusive gateways. The desired level of enforcement is determined by sets of validators.

Three levels of sets have been defined: public, private, and user-defined. The public set mandates that decisions are verified by a larger set of nodes compared to the participants involved in the process, while the private set requires that all process participants validate the decision taken. Lastly, the user-defined annotation enables the modeler to specify a set of participants responsible for verifying the decision. In Table 2.2, it is shown graphically the three levels of sets related to the enforceability of decisions.




Symbol	Level of enforceability of decision
	Public
	Private
	User defined

Table 2.2: Enforceability of decisions

Figure 2.3 shows the representation of the contract presented in Figure 2.1 enriched with SecBPMN2BC notations referred to properties as privacy and enforceability of decisions. As visible, the data object "Pathology documentation" is defined with a privacy sphere of type private, thus the content of data object has to be owned by all participants in the process: Company Employee, who is the first one to introduce it into the collaboration diagram, Health Care Fund, which is going to access it during the evaluation of the request, and Medical Office that will read the document in case of positive answer from Health Care Fund.

Analyzing the other data object present in the collaboration diagram, "Available dates",

it is linked to a privacy sphere of type strong-dynamic. It means that, in that particular moment, the only participant who has to own the data object is the one that is certain to access that data, and it is represented by Company Employee who is going to access it in the task "check available dates". The method has to be able to guarantee the sharing of data objects among all and only participants who need them before the activities in which they have to effectively access them.

Always in Figure 2.3, for what regards the exclusive gateway into the Health Care Fund pool, the enforceability of decision property associated with it is set at level private: each participant to the process has to validate the decision taken by Health Care Fund about the compliance of request by Company Employee. The decision will be made by taking into examination the Pathology documentation contained in the data object, which is examined in the activity named "Check compliance". The method developed, to ensure that the validation can be performed by all participants, has to guarantee that each validator owns their copy of data upon which the decision is taken before the validation phase.

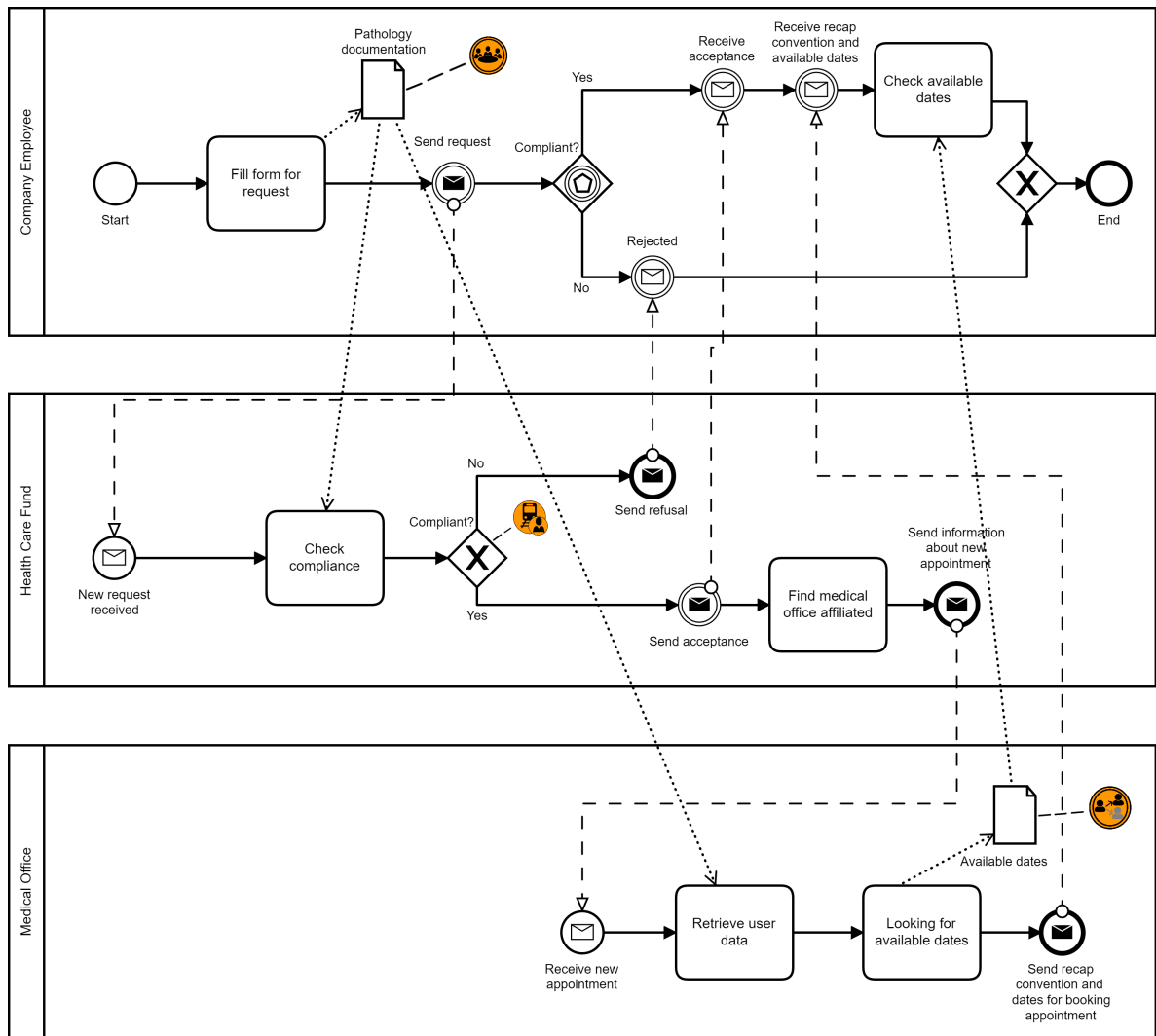


Figure 2.3: Running example with SecBPMN2BC notations

Since SecBPMN2BC has been structured for BPMN Collaboration, the initial part of our work is aimed at translating and adapting these security concepts in BPMN Choreography proposing a method to enforce security properties.

2.4. BPMN Choreography

BPMN Choreography is a modeling technique used to represent the interactions among multiple participants or organizations in business processes. It focuses on the exchange of messages, to highlight how multiple parties cooperate and coordinate their actions to achieve a shared business goal.

A choreography diagram is the set of tasks, messages and gateways used to represent a

process. Choreography tasks refer to the actions or activities performed by participants. They capture the activities that are triggered by incoming messages or events from other participants while they don't focus on internal details of each participant's behavior.

Tasks illustrate the sequence of activities in the process encapsulating the logical flow of tasks based on the messages exchanged between participants. Successive tasks are connected by sequence flows, i.e. an arrow, that determines the sequential nature between two or more tasks. In a task there is always an initiator, i.e. the participant on the high side of the task, and at least one receiver, i.e. the participant on the low side of the task, with the possibility to have communications unilateral or bilateral: in case of tasks that don't require an answer from receivers, the message is sent only from initiator side and the communication is defined unilateral, in other cases, there are messages from both initiator and receivers and the communication is identified as bilateral.

An instance of both cases is reported in Figure 2.4.

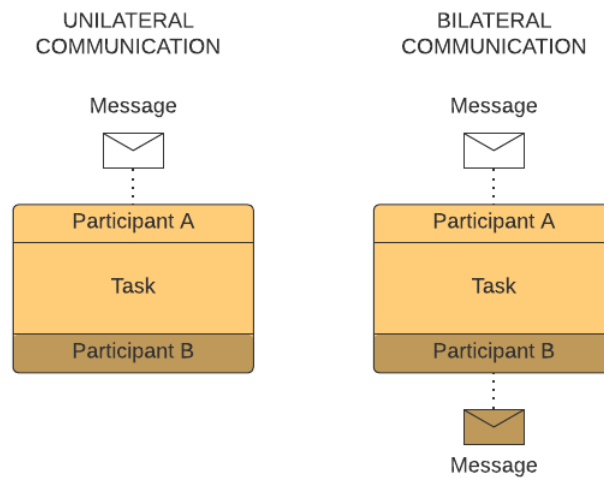


Figure 2.4: Unilateral and bilateral communication

A general rule in the definition of successive tasks is that, after the first task which is rule-free, a task initiator must be involved in the prior task as initiator or receiver.

Messages represent information exchanged between participants and describe the flow of data, requests, or notifications from one participant to another. They are characterized by asynchronous communication, thus the sending participant does not wait for an immediate response from receivers: this feature allows participants to continue their activities

independently while the message is being processed by receivers. The content of message is not specified directly from the modeling language and they can represent various types of information, data, requests or notifications that are relevant to the process.

Gateways, depending on its nature, can represent decisions based on particular data exchanged or generating alternative or parallel paths for that process. The absence of a centralized mechanism for data visibility and evaluation imposes limitations on how gateways can be utilized in relation to the choreography activities that come before and after them. For each type of gateway there are specific rules described in the official document in [28].

In case of exclusive gateways data used to take decision must have been contained in a message sent in the previous tasks. Each participant affected by the gateway must have sent or received the message with data upon which the decision is taken. Moreover, one or more participants have control of the gateway and effectively are responsible for decisions made and they are represented by initiators of tasks immediately after the gateway. These initiators must have received or sent the messages with data upon which the decision is taken and even the receivers of tasks after gateways should have the same understandings of data, which means that they must have received or sent messages with data upon which the decision is made, otherwise, they can't expect a message from the decision in that point of the diagram. Parallel gateways represent a point where multiple tasks can occur in parallel and only where both branches of gateway have completed their execution, the task after the conjunction parallel gateway can start. In this case, there aren't restrictions to follow, it is necessary to apply the general rule of two successive tasks. Other types of gateways such as inclusive gateways, event-based gateways and complex gateways are representable in choreography diagram, however, due to the purpose of our work, the focus will be primarily on exclusive gateways and the security properties that directly concern them.

A demonstrative example of choreography diagram that summarizes all concepts introduced can be found in Figure 2.5. It is reported as a process that represents the interactions between a Customer and a Library Online System to borrow a book that has to be contained and picked up in the physical store to be booked successfully. The process starts with the customer who accesses the website of the library with proper credentials and then sends the request for the loan of a particular book. In this case, the book can be present in the library store and, for this reason, the booking is accepted and is set a date for picking up the book. Otherwise, the loan request is rejected by the system.

We decided to show the choreography of running example defined in Section 2.2 in Chapter

3 because the translation of a process from collaboration to choreography is a part of our method developed to enforce security properties through automated procedures.

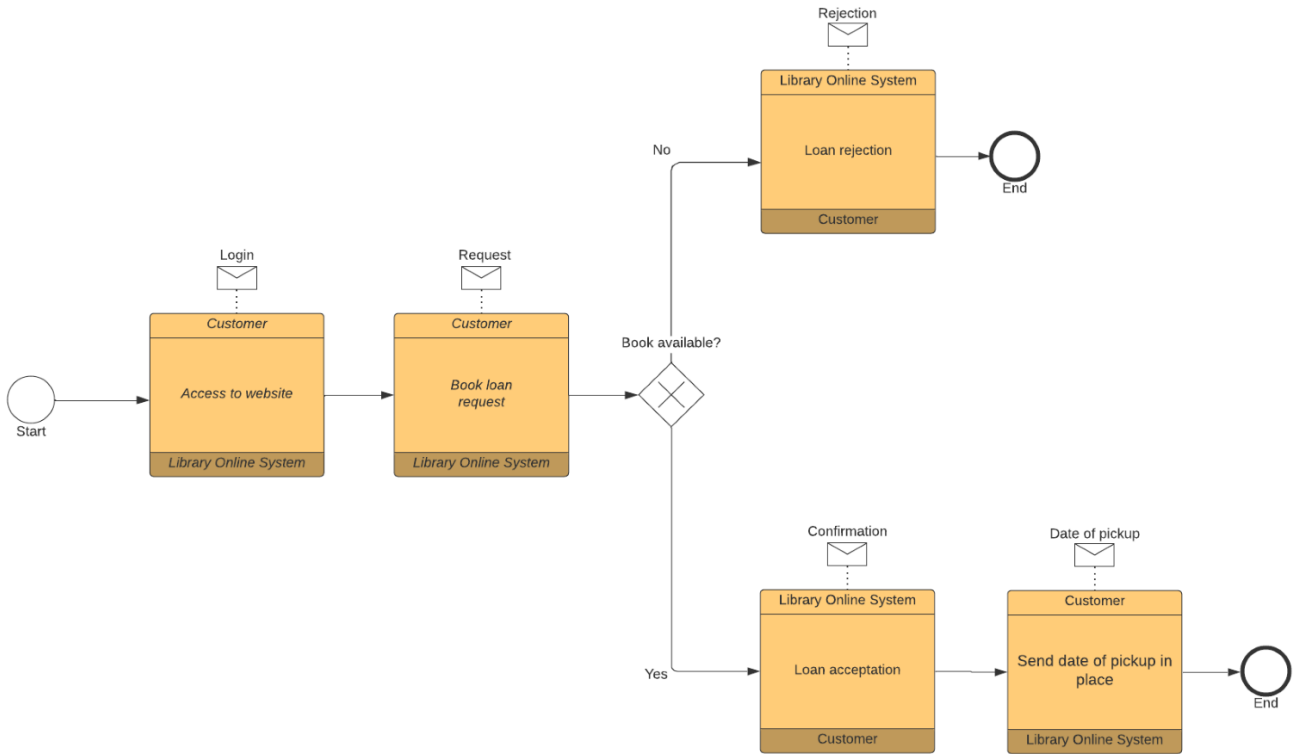


Figure 2.5: Choreography diagram of book loan

2.5. Corda

Corda is an enterprise-oriented permissioned blockchain developed by R3. It acts as a decentralized, open-source ledger technology that enables secure and confidential transactions among participants.

One of Corda's important features is privacy, ensuring that sensitive information is shared solely between relevant parties while adhering to regulatory requirements. It utilizes smart contracts, referred to as CorDapps, to govern the behavior and interactions of shared data within the network. Corda also focuses on significant emphasis on interoperability, facilitating the integration with existing systems and databases. The general purpose is to provide trust, efficiency, and transparency in business transactions while upholding confidentiality and privacy. The platform allows the execution of contracts by providing a framework to implement business processes and facilitates the creation, execution, and

transaction of legally binding agreements through the collection of required approvals and signatures.

Corda network is a private network that can be seen as a fully connected graph where each peer, known as a node, potentially has the possibility to interact with other peers. No information are shared in broadcast among all nodes, instead, the approach is based on a "need-to-know" basis, thus only nodes with permissions are going to access specific data shared.

Participants of Corda network refer to the entities or actors that join the network and engage in transactions. These participants can be individuals, organizations, or even automated systems. Each participant on the Corda network possesses a unique cryptographic identity, which ensures secure and authenticated interactions. Participants can create and manage states, propose transactions, and participate in the consensus process to validate and agree upon the shared ledger.

States represent shared facts or agreements between participants on the network and they can represent various types of information, such as financial assets, ownership records, or any other data that needs to be shared and recorded on the ledger. Since there is no central ledger, not all nodes know all states and each node has its own vault of states which stores shared states related to that node. Figure 2.6, obtained from [31], shows an example of states shared between two nodes.

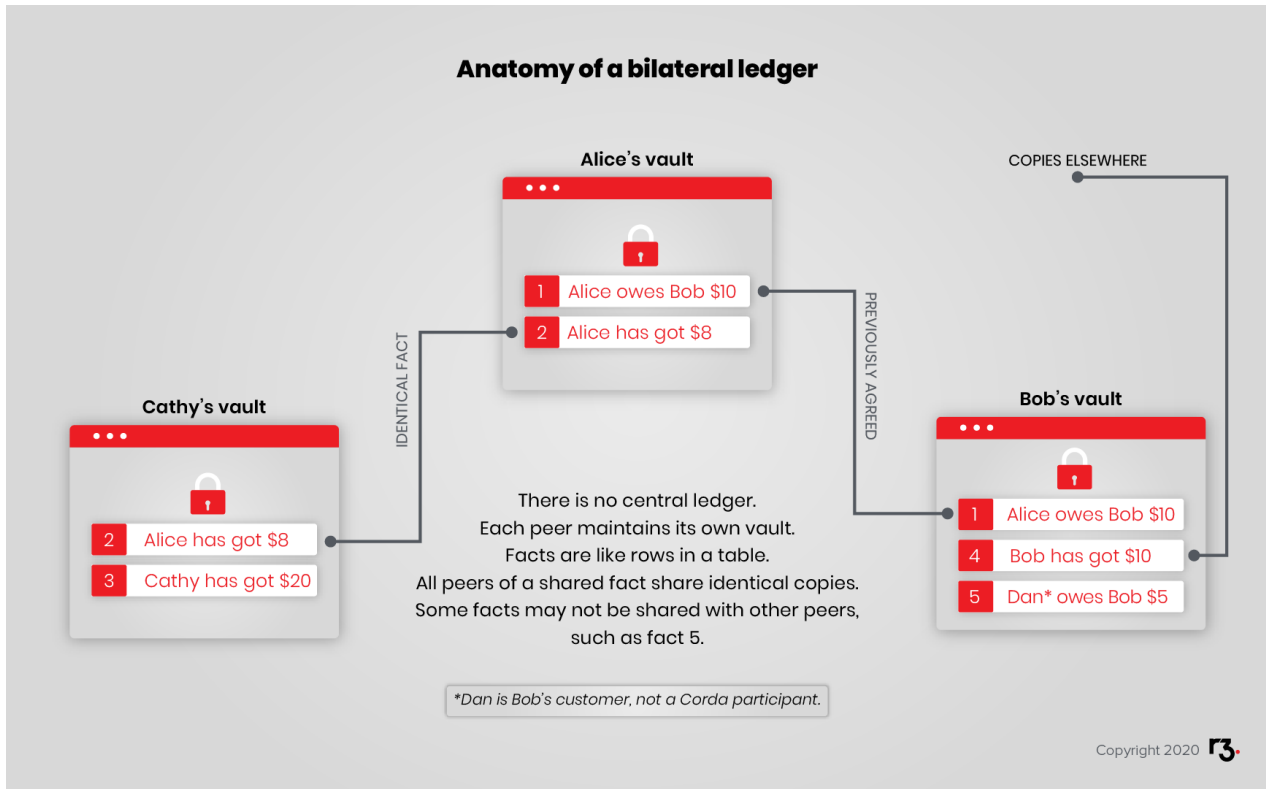


Figure 2.6: States shared between nodes of Corda network from [31]

Transactions represent proposed changes to the shared ledger state: more in detail, they are a data structure that encapsulates a set of input states, output states, commands, and other relevant information related to the proposed change in the ledger. Transactions in Corda consume existing states, i.e., input states, and produce new states, i.e., output states. Input states represent the current state of the ledger, while output states represent the desired state after the transaction is completed. Among the relevant information in a transaction, attachments refer to additional files or data that can be associated with a transaction. They are hashed using a cryptographic hash function, i.e., SHA-256, then stored in the transaction and transferred to the counterparties.

Contracts define the rules and logic for the behavior of states on the network, by specifying rights, obligations, and constraints associated with the states involved in a transaction, ensuring their validity and enforceability. Contracts verify the validity of transactions by checking the input and output states, commands, signatures, and other relevant conditions, ensuring compliance with defined rules and preventing unauthorized or invalid transactions. They act as an agreement enforcer, ensuring that all parties involved in a transaction adhere to the predefined terms and conditions.

Timestamps define that something happened in a certain window in time. They are

helpful in order to assign a deadline for some actions and even for the verification of the action itself. For instance, if there is an offer related to some goods and the answer to that proposal comes after the deadline specified, it won't be accepted.

Commands encapsulate the intention of a transaction, collecting the input and output states and specifying the list of required signers. Commands are associated with a specific contract and interact with the contract's code to enforce the rules and constraints defined by the contract ensuring that transactions comply with the contract's conditions. Figure 2.7 describes the function of commands.

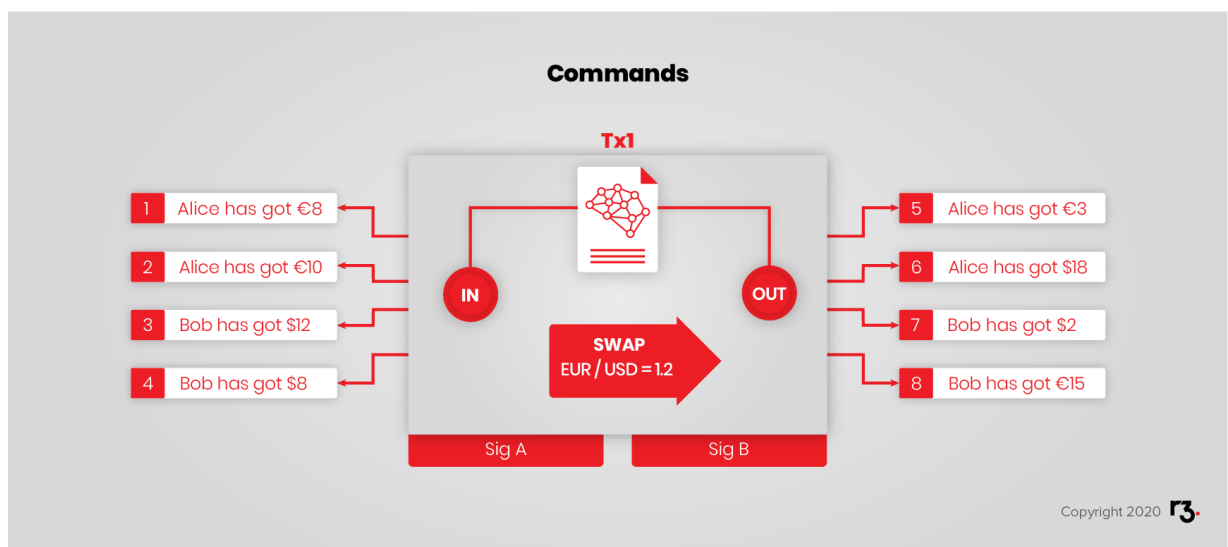


Figure 2.7: Commands in transactions from [31]

Notaries act as a trusted authority responsible for validating the uniqueness and authenticity of transactions. It verifies that a transaction doesn't contain double-spent states and that the inputs used in the transaction have not been consumed in any other valid transaction. After the validation by notaries, input states are marked as historic to avoid double-spending issue. A Notary is present by default in the network.

Flows are responsible for the evolution of contract states by proposing, verifying, and executing transactions. They coordinate the interaction between multiple parties involved in a particular business process or transaction by defining a sequence of steps and actions that participants need to perform to progress states. Flows are designed to be asynchronous and non-blocking, allowing nodes to continue other activities while waiting for responses, improving efficiency and enabling parallel execution of multiple flows.

CorDapps are distributed applications built on Corda platform that can be programmed with different programming languages like Java, Kotlin and C#. In this work, the lan-

language chosen is Java: a reason for using Java in Corda development is its versatility and widespread adoption, with a large community of developers and a rich ecosystem of libraries and tools.

3 | Method definition

This chapter describes the method developed for the enforcement of security properties by generating and deploying contracts in the Corda blockchain. In Section 3.1, there are described all steps necessary for the method realization. Section 3.2 provides a list of assumptions adopted to formulate the model for what regards processes considered, elements of choreography diagram and Corda object properties. Section 3.3 details how to extend BPMN Choreography modeling language with security notation and innovations introduced to manage such properties.

3.1. Method description

Our work focuses on the development of a method to guarantee the enforcement of security properties such as the enforceability of decisions and data privacy, more details are reported in Section 2.3, through the creation of contracts in Corda blockchain.

The method described below, with stages detailed in Chapter 4 and 5, represents the main part of our contribution. It proposes a new way to enforce the security properties of smart contracts modeled from business processes. The first aspect to analyze is the representation of the contract we decided to describe: it is depicted in BPMN Collaboration specifying activities it is composed of and, then, it is enriched with security properties of privacy and enforceability of decisions according to SecBPMN2BC. The method, therefore, receives a collaboration diagram enriched with security notations and performs the following steps until the realization of contracts:

1. The first stage is related to the transformation from collaboration to choreography diagram: it consists of an initial phase where the first adaptation between collaboration and choreography takes place without considering the defined security properties and the existing data objects, based on the interactions between the participants. In this way, the resulting diagram is a foundation that will be enriched by the elements not initially considered. Therefore, the next phase involves representing the security properties and data objects in the choreography diagram, aiming to

depict the same scenario as the original collaboration diagram. The crucial aspect is to maintain the meaning represented by the security properties and not disrupt the order of activities that were performed. Taking a higher-level perspective, it won't be possible to describe all the individual activities in detail as they are presented in the collaboration diagram. Instead, the focus is on the interactions, motivated by the contracts that will be implemented in Corda, which are also based on the interconnections between the participants. After that, as the last phase of this stage, the diagram is modified in order to reach a new choreography diagram without the graphical notations of security properties or data objects, but with additional tasks that are going to incorporate and apply the meaning of security properties.

2. The second stage is represented by the transformation of choreography diagram into contracts in the Corda blockchain. This transaction is composed of two phases: the first one is the definition of conceptual mapping between choreography diagram and contracts which aims to specify how each element in the diagram will be represented by a corresponding element in the Corda contract; in the second phase, instead, the transformation rules will be defined, which are algorithmic procedures that automate some aspects of creating individual elements, defining their names and attributes. It is important to emphasize that the transformation rules are based on conceptual mapping and add a level of depth, showing the actual characterization of the individual elements.
3. The last stage consists of the effective implementation in Java programming language of the elements of CorDapp produced in the previous stages.

This method doesn't aim to provide a fully automated generation and execution of CorDapps, because it is necessary the human contribution to adapt the skeleton of CorDapp with all the requirements of a Corda contract and with information related to possible files or messages exchanged in the contract.

3.2. Assumptions of method

Before starting with the initial phase of the method, an explanation of the assumptions made during the development of each phase of the method is necessary.

For what regards the type of business processes considered, we have chosen to delve into interaction-rich processes because they are better suited to the entire transformation process that follows: the choreography diagram represents tasks based on interactions between participants, and Corda contracts primarily focus on transactions that describe

agreements among multiple participants, as described in Chapter 2. For these reasons, we find it more effective to analyze processes with multiple interactions rather than those with few interactions.

In this scenario, it has to be explored the importance of protecting data exchanged among parties. Commonly, distinct types of data states are data sent by a party to others into a private network (i.e., data in transit) and data stored permanently that does not travel in the network (i.e., data at rest). In most cases, these types of states are treated separately and with different techniques of encrypting data, to ensure a higher level of security to both. In this work, there is no need to make this distinction and to analyze the different states of data: the focus will be on data exchanged among participants through the interactions depicted in the choreography diagram.

Regarding the realization of contracts in Corda, we need to specify the following premises:

- Each node of the network is characterized by an identifying name, such as the company name, a location and the country where it operates, for instance, the city and country of the headquarters. These three attributes, i.e. "Organization", "Location" and "Country" are specified in the build.gradle file in CorDapp and will be able to identify the individual nodes in the evolution of the process.
- The presence of a single state that evolves over time through various transactions: the contract regulates the evolution of a state, which is modified in each transaction by acting on its attributes. A choreography diagram is therefore represented by a state and the related contract in Corda.
- Fixed attributes for the state are the ones related to the initiator and receivers of the workflow and the id of the state which is generated once and then associated with input and output states.
- Each transaction changes a specific attribute in the state, thus we are going to have an attribute related to each task of the choreography diagram. If these tasks are related to security properties, specific attributes are generated, as detailed in Section 3.3. For other tasks, an attribute that describes the actions performed in that step will be added.
- In the translation from the choreography diagram to Corda, when the first task of choreography is mapped into a workflow with the related transaction in Corda, this specific transaction will not have the input state since it is the first one of the list, but it is going to produce only the output state.
- When a workflow is generated, automatically they are added an initiator flow and

a responder flow related to the initiator and receivers of the corresponding task.

3.3. Extended BPMN Choreography

BPMN Choreography modeling language doesn't provide the representation of data objects and security properties such as privacy and enforceability of decisions. To address this limitation, we have considered including in the choreography diagram the security notations already presented in collaboration derived from the SecBPMN2BC approach.

Several rules were applied to represent these properties in the choreography diagram without distorting the intrinsic meaning of each property, aiming to depict a situation as similar as possible to the original one. Referring to the enforceability of decisions, since it is linked to the exclusive gateways present during the initial adaptation, we chose to maintain the notation as found directly in the collaboration diagram. In this case, we can therefore state that the property remained unchanged and carries the same meaning because, in the choreography diagram, there will be the same participants as the collaboration one: the set of validators of decisions can be confirmed. The exceptions admitted are linked to the presence of public level enforceability of decision: in this case, the property will be changed to a private one referring to all participants of the contract since the inclusion of external participants is not admitted in Corda.

As for the data objects, we hypothesized representing them by maintaining the same graphical notation replacing messages or next to the message itself, depending on the presence or absence of another message from the same participant in the task. The choice of the task where data objects are represented is made by trying to keep unchanged the participant who first introduces the data object in question, as well as the order in which other process participants will access it in subsequent tasks. The same symbol will be used to emphasize the privacy sphere associated with the data objects, with the exception of the public level which becomes private for the same reason of enforceability of decision. Based on interactions, we are aware that not all situations will be replicable, but our goal is to adapt and model the diagram in a way that ensures the representation of these security properties.

The running example is represented through a choreography diagram enriched with security properties in Figure 3.1.

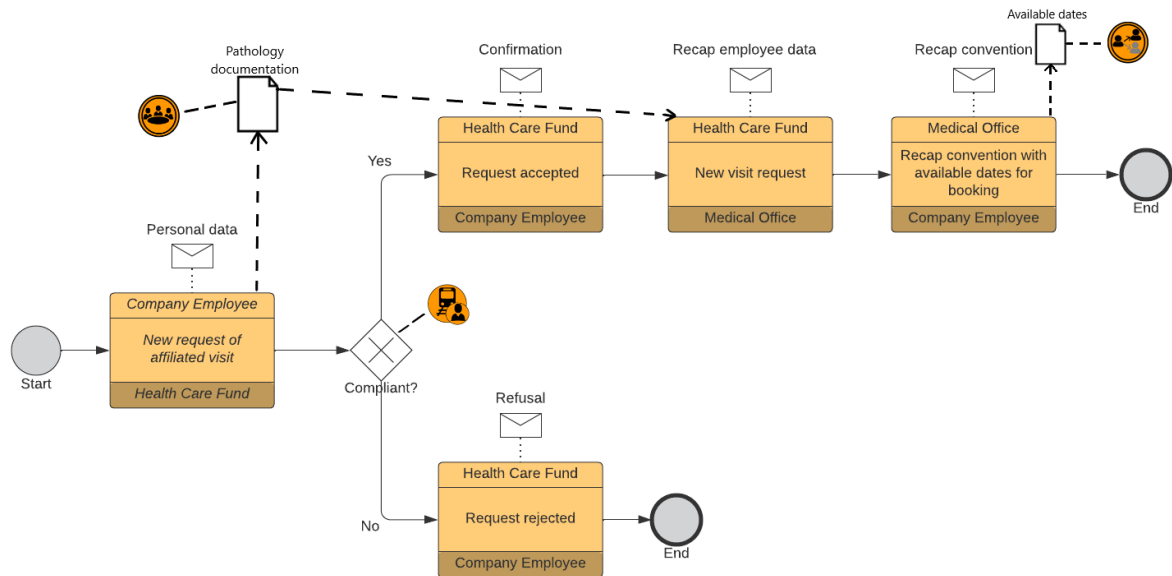


Figure 3.1: Running example choreography diagram enriched with security properties

Looking at Figure 3.1, it can be noted how the security properties have been graphically added to the diagram: in the task "New request of affiliated visit", in fact, there is a reference to data object "Pathology documentation", indicating its addition to the message containing personal data of the Company Employee. Moreover, the arrows linked to data object indicate that Company Employee is the participant responsible for introducing the "Pathology documentation", inheriting this characteristic from the collaboration diagram in Figure 2.3. The task where its content will be accessed is the task "New visit request", where Medical Office will become aware of the patient's personal data and will analyze the documentation related to their pathology. The privacy sphere associated with it remains the same as the one in Figure 2.3.

The other data object represented in the diagram can be found in the task "Recap convention with available dates for booking", where the data object "Available dates" is represented near the message of recap convention. The incoming arrow of the data object indicates that the content owner is Medical Office, which will be the first to introduce the data object in the choreography diagram. The privacy sphere defined for the data object is of a strong-dynamic type, inherited from the collaboration diagram in Figure 2.3. Furthermore, it can be noticed that it is not connected with outgoing arrows to any other task. This means that no other participant apart from Medical Office and Company Employee should own that data, otherwise, the privacy property would be violated.

The exclusive gateway in the diagram is enriched with the security property of enforce-

ability of decision without changes with respect to the one in Figure 2.3. The diagram is ready to go further in the next stage where choreography diagram will be modified in order to handle the security properties just added.

In order to address the enforcement of decisions, the aspect to analyze is how to carry out the validation of the decisions taken: as described in Section 2.4, the evaluation of gateways is based on data exchanged with messages, thus all validators must own the data used to make decisions to be able to validate them. Validation is performed ex-post by a set of validators, i.e. receivers of choreography tasks, specified by levels of enforceability of decision, as reported in Table 2.2.

By design of BPMN Choreography, receivers of the first tasks after the gateway own the data used to make the decision: for this reason, they are going to represent a subset (or total set) of validators. Since other validators may own or not have data used for decision-making, it is convenient to separate these types of validators, by defining two categories.

The first category of receivers will be composed of receivers of the first tasks after the gateway, because we are sure that, at that point, they own the data upon which the decision is made, otherwise, they would violate the choreography constraints.

On the other side, validators of second category are participants that are not included in the first tasks after the gateway, but they are in the set of validators required by the level of enforceability expressed. In this case, there could be two separate cases: validators already own data used for decision, they are validators of second category without priority; the other case is represented by validators that don't have data used to make decision, they are called validators of second category with priority.

Since all validators of first and second category must own data used for decision to proceed with validation, the only category that needs an additional Choreography task to be aware of that information is the second one with priority. In this case the additional task will be added exactly before the gateway to allow the sharing of data, while the validation will be performed for both validators of second category with and without priority in an additional task after the gateway. The initiator of additional will be in each case responsible for decision. With regard to validators of first category, they will perform the validation in the tasks already defined for them.

Additional tasks for sharing decision data will contain "Decision_sharingData" while additional tasks related to validation performed by validators of second category will hold "DecisionPath_Assessment" in task name.

As stated in Section 3.2, data objects are going to be processed as content of messages in choreography diagram. In graphical representation, we have a data object that enters in the diagram in a specific task and the initiator of that task is the one that will handle that data object for the first time. Moreover, data object is connected by dashed lines to other tasks where it is going to be accessed by other participants, based on the privacy sphere associated to that data object as described in Section 2.3.

In order to manage the presence of data objects and share the message containing data object with all participants that are in the set specified by privacy sphere, there is needed an additional task where the message exchanged will contain data object and it will be placed after the first one where data object enters in the diagram.

By analyzing the first task where it is managed, we are sure that the receivers of that task are included in the set of participants that have the right to access that data object. Furthermore, in creating the additional task we have to consider that initiator of the additional task has to be the initiator or receiver of the previous task by design, thus the additional task will include at least that initiator and receivers. After that, selecting the receivers to be added to that task there can be two cases: the initiator and receivers of the previous task are all and only participants that have to access that data object, in this scenario, the initiator and receivers of the new task remains the same; otherwise, initiator and receivers of the previous task are not all and only participants that have to access that data object, in this case the initiator of the task will be the same, while receivers set will contain receivers of the previous task and other participants specified in privacy sphere that are going to access data in the future tasks of the process. Additional tasks for sharing data objects will contain "Privacy_sharingData" in the task name.

Before proceeding, it is needed to define an order to follow during the evaluation of these properties when they are both present in the choreography diagram. The privacy property related to data objects is evaluated first, given the possibility that two subsequent tasks may require the use of that data object. Therefore, the presence of a data object is immediately handled by the method, generating the additional task, if necessary. After evaluating the first property, the presence of exclusive gateways and the related level of enforceability of decisions is analyzed using the mechanism described earlier. An example of application, starting from the enriched choreography diagram in Figure 3.1, is depicted in Figure 3.2.

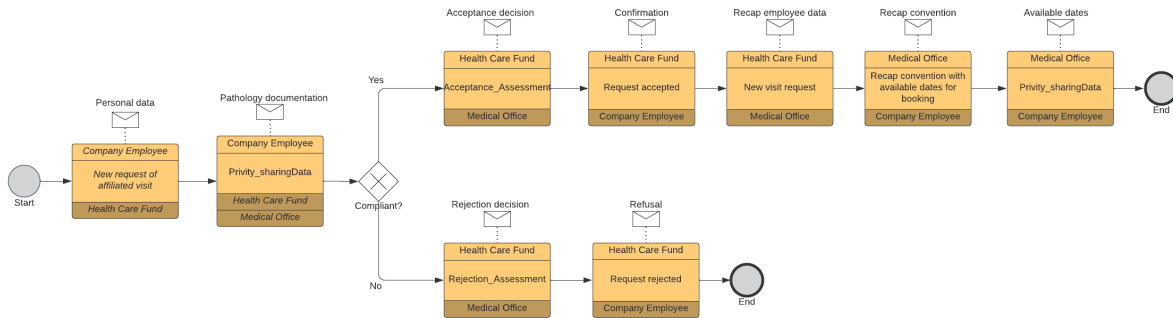


Figure 3.2: Choreography diagram generated from extended choreography diagram in Figure 3.1

Figure 3.2 shows a BPMN choreography diagram generated by the rules described in Section 3.3: at the beginning, it is checked if there are present any data objects and, if positive, it is analyzed the related privacy sphere. As represented in the choreography diagram, data object "Pathology documentation" is depicted in the first task of the diagram and is managed by Company Employee, while data object "Available dates" enters the last task of the branch activated after the acceptance of the request and is managed by Medical Office.

Starting from the first data object, i.e., "Pathology documentation", we can observe that Company Employee and Health Care Fund are participants of the task where the data object is introduced for the first time and, thus, they belong to the set of participants that have to own that data. Since the outgoing arrow point out the task where it is present Medical Office as initiator, the privacy sphere associated with the data object is of type strong-dynamic: this means that even this participant has to be added to the set of participants that have to own the data object. Applying what is described in Section 3.3, it is added a new task in the choreography diagram after the first one, named "Privity_sharingData", characterized in the following way: as initiator the same initiator of the previous task, i.e., Company Employee, as receivers both Health Care Fund and Medical Office and as message sent from the initiator to receivers "Pathology documentation". In this way, it is ensured that all participants in the set defined by the privacy sphere are going to own the data object in order to continue the execution.

Analyzing the second data object, i.e., "Available dates", it is introduced in the last task of the positive branch and the participants of that task are Medical Office and Company Employee, who will definitely be part of the set of those who must own that data object. In fact, examining the privacy sphere associated with the data object, it is a strong-dynamic one and no other participants are requested to own that data because only Company

Employee will use that data at that point in the diagram. For this reason, it is added a new task in the choreography diagram after the "Recap convention with data available for booking", named "Privity_sharingData", with the same initiator and receiver concerning the previous task.

Continuing with the process, the presence of a exclusive gateway is checked, and it is determined on which data, exchanged in previous tasks, will be based the decision and which participant will be responsible for making it. In this case, the decision will be explicitly based on the data object "Pathology documentation" and Health Care Fund, the initiator of both tasks immediately after the gateway, will determine whether Company Employee meets the requirements to benefit from the conviction. At this point, the enforceability level of the associated gateway is taken into consideration. In this case, the level is set to private, which means that all participants in the process must take part in the validation phase of the decision made: specifically, Health Care Fund is going to be a validator of first category, based on the presence on the first tasks after the gateway, and Medical Office is going to be a validator of second category.

To validate the decision, validators must have their own copy of the data on which the decision is based. Taking into consideration the added task used for managing the privity property, it can be observed that at the time of the decision, both Health Care Fund and Medical Office already possess the "Pathology documentation" data on which the decision is based, and meanwhile, it has not been modified by other tasks. Therefore, in this case, there is no need to add another task for data sharing before the gateway because the validator of the second category is without priority.

In order to validate the decision, as specified in Section 3.3, two new tasks are added after the gateway, one for each branch, where validators of second category can perform the validation: if the branch selected by Health Care Fund is the one related to acceptance, the task called "Acceptance_Assessment" is added, where the initiator is Health Care Fund, the receiver is Medical Office and the message contains the acceptance of the request. On the other hand, if the selected branch is the one related to rejection, the task called "Rejection_Assessment" is generated, with the initiator and receiver remaining the same and the sent message will contain the rejection of the case.

In this way, it is performed the validation by the validator of second category, while the validator of first category is going to validate the decision in the respective second task in each branch with the already defined tasks. The rest of tasks is left unchanged.

4 | From Collaboration to Choreography

This chapter details the first mapping of the method proposed in this thesis. After having defined the collaboration diagram of our contract, we have to represent it by using choreography modeling language. Section 4.1 presents guidelines that indicate how to transform a collaboration diagram into a choreography one. Section 4.2 shows the practical transformation from collaboration to choreography diagram based on the running example.

4.1. Collaboration to Choreography transition

Guidelines adopted during the transition from Collaboration to Choreography standard are represented in Table 4.1.

Collaboration element	Choreography element
Pool/Lanes	Participants (initiators and/or receivers)
Start/End events	Start/End events
Message events	Tasks
Message flows	Initiator and receivers
Tasks	No direct mapping
Exclusive gateways	Exclusive gateways
Other types of gateways	No direct mapping
Sequence flow	Sequence flow
Data objects	No direct mapping

Table 4.1: Mapping Collaboration to Choreography

From pool/lanes to participants

This association in Table 4.1 describes how to define the participants of the choreography diagram starting from the collaboration one: the process consists of selecting pools and,

if present, lanes, to have a frame of all possible participants in the choreography diagram. Lanes are treated as different participants because if we consider only pools it could be difficult to represent elements like gateways or specific messages among participants and we would increase the granularity level hiding other information. Since the choreography diagram aims to map the interactions among participants, we are sure all the pools and lanes will be included in the choreography diagram as initiators or receivers of tasks.

From start/end events to start/end events

Table 4.1 describes how start and end events are mapped into choreography diagram. Based on the logical flow, the event that starts the process and the one linked to the last interaction are present in the choreography diagram. Regarding intermediate start and end events, they can be present or not based on the link with interactions. If an exclusive gateway triggers two branches that lead to two end events, each one with interactions among different participants, in that case, both events are reported.

From message events to tasks

Since in the choreography diagram the focus is on the interactions among participants, the central mapping is based on the presence of messages in the collaboration diagram that, in the mapping process, become choreography tasks as reported in Table 4.1.

In collaboration diagram for each message event inside a lane/pool that sends the message, there is a message event that receives it in another lane/pool. In the transition process, the pair of message events, i.e., sender and receiver, is mapped with a choreography task, visible through the identification name, which has to summarize the meaning of that exchange of messages among the participants. Since in choreography diagrams there are strict rules in the definition of the initiators of tasks, as explained in Section 2.4, the tasks present at the end of the process can be not only the ones who are defined by message events: there may be other tasks that represent a communication which doesn't modify the sense of the diagram but they are necessary to be compliant to the generation rules of choreography tasks.

From messages flow to initiator and receivers

In order to link message events among participants in collaboration diagram there are needed some connection lines that connect the sender and the receiver. This message flow, which takes into consideration the pool or lanes of sender and receivers, provides information about the initiator and the receivers of the task in choreography diagram, as reported in Table 4.1. Once the choreography task has been generated by the message events, there is the necessity to assign the roles of initiator and receiver of that task based on the flow of connection lines of message events in the collaboration diagram. In order to address this point, the pool or lane of the sending message flow is the initiator of the

task, while the pool or lane of the receiver of the message flow is the receiver of the task.

Tasks not directly mapped

As reported in Table 4.1, not all collaboration tasks are directly mapped into collaboration diagram. This is mainly due to the different levels of granularity to which the two models refer: in the collaboration diagram, there is a pool or lane that contains individual activity performed by a specific participant to reach a certain business situation, without this being known to other participants; instead, in the choreography diagram, we represent the actual interactions, without going into the detail of the individual activities performed by each participant. With this premise, the tasks representing activities internal to the individual organizations will not be represented in the choreography diagram.

Exclusive gateways conservation

As reported in Section 3.2, exclusive gateways are the type of gateways we are interested in due to their link with decisions. Exclusive gateways are mapped in the same way moving from collaboration to choreography diagram as reported in Table 4.1. This mapping is made in order to keep the decisions even in the choreography diagram, respecting the sequence of events that the decision influences. The exclusive gateways in the choreography diagram are then enriched with the presence of security property of enforceability of decisions as detailed in Section 3.3.

Type of gateways not mapped

As reported in Table 4.1, other types of gateways are not mapped: gateways that are not linked with decisions, such as the parallel gateway, and that involve tasks linked to internal activities related to a specific participant without interactions with others are not mapped. Furthermore, there is the possibility that the sense of a particular gateway is already included in the choreography task: this is the case of an exclusive gateway in a pool that refers to the decision which has to be taken and an event-based gateway in another pool which path is linked with the decision just taken. In this case, in the choreography diagram we will not have the representation of both gateways, only the exclusive gateway with choreography tasks that describe the possible paths as initiator the participant that is responsible for the decision and the other participants involved.

Sequence flow conservation

The sense of order and sequentiality in collaboration diagrams is given by a sequence flow represented through arrows that link tasks. This type of mapping, as reported in Table 4.1 is conserved as the choreography diagrams exploit the same way to indicate the sequence in which tasks have to be performed.

Data objects not directly mapped

As detailed in Section 3.2, data objects depicted in the collaboration diagram are not directly represented in this phase of transition from collaboration to choreography, as visible in Table 4.1. They are not included in this transition process for a dual reason: the choreography standard does not handle the representation of these types of objects and there is uncertainty related to the presence of the task in which it is represented in the collaboration diagram. They will be included in the extension of the choreography diagram with the presence of security property of privacy as discussed in Section 3.3.

4.1.1. Implications

Implications derived from mapping rules:

- There is not a unique translation of a collaboration diagram into a choreography diagram: since the focus is on the interactions, we can argue that all messages must have a corresponding task in the choreography diagram; for the other elements to be included, their presence may depend on what you want to represent and where to put the emphasis. A brief example can be done if we imagine two tasks in a collaboration diagram performed in parallel by the same participant and, in both tasks, there is a message sent to the same receiver: one possible choice can be to represent a single task in choreography diagram with initiator and receiver and a unique message sent which takes both the information of single messages; on the other hand, it is possible to generate two choreography tasks that execute in parallel and send messages. Both situations are feasible, for this reason, we act to preserve the general sense of the contract.
- Not all the situations described with the collaboration model are reproducible with the same level of granularity in the choreography diagram: in the former, there is a description with a low level of granularity, potentially going to represent all the single activities executed by a participant, while in the latter one, the focus is on the interactions between participants.
- Gateways not influenced by a decision are not reported in translation: as described in Section 3.2, since we have to enforce specific security properties such as the enforcement of decisions, all gateways unrelated to that property are not reported in the mapping if they are not fundamental to preserve the constraints of the contract.
- General flow of contract over the single task performed by organizations: in this way, we are not strictly related to the representation of internal tasks, on the contrary, we focus on the general flow and sense of the contract in order to respect the sequence of interactions among participants and related constraints.

4.2. Application of mapping from Collaboration to Choreography

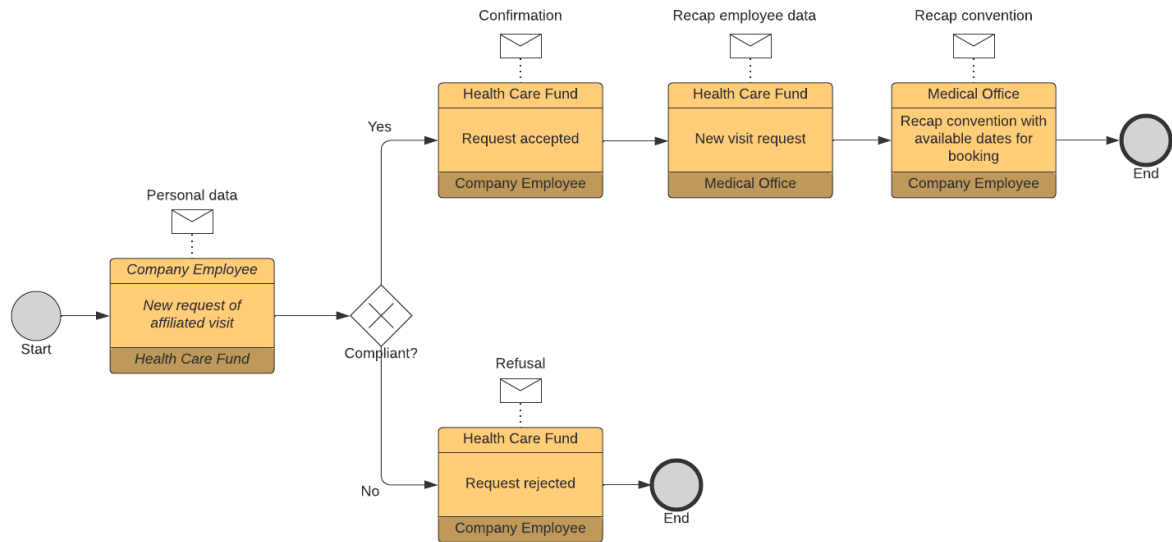


Figure 4.1: Choreography diagram of affiliated medical visit

Figure 4.1 shows an application of the choreography diagram generated by guidelines described in Section 4.1 starting from the collaboration diagram represented in Figure 2.1.

If we collect all the initiators and receivers from choreography tasks in the diagram, they are the same participants depicted by pools in collaboration diagrams: Company Employee, Health Care Fund and Medical Office. There is a start event at the beginning, while the end events are one for each branch executable after the exclusive gateway.

Regarding tasks generation, it is visible that for each pair of message events, used for sending and receiving messages, a choreography task is created: for instance, the message events that exchange the message from Company Employee to Health Care Fund containing personal data of employee are represented with the choreography task "New request of affiliated visit". Regarding the initiator and receiver of tasks, they are selected based on the flow of the message events: the participant that sends the message will represent the initiator of that task, instead the receiver of the message will be the receiver of the task.

As detailed in Section 4.1, not all the tasks are represented moving from collaboration to choreography diagram: if we observe the single activities performed by each participant

in the collaboration diagram, we notice that tasks that describe personal behavior such as "Fill form for request" or "Check compliance" are not present in the choreography diagram. This highlights the different levels of granularity on which the standards are built. In the choreography diagram the personal activities are "hidden" behind the message sent: analyzing the choreography task "New request of affiliated visit", we can imagine that the personal information sent by Company Employee are the result of small tasks computed such as retrieving personal id card and fill a form that arises in the message sent. In this way, we are not exploring how the single activities are executed by participants, but we are sure that they have been done in order to send the message.

When comparing the two diagrams, a difference can be seen in the treatment of the gateways. While in the collaboration diagram if we count the number of gateways we obtain as total number of three, whereas in the choreography one, we have only one gateway. Following what is described in Section 4.1 about the gateways in the mapping, we can ascertain that the mapping is performed in the following way: the exclusive gateway in the Health Care Fund pool influences the event-based gateway in the Company Employee pool, for this reason, this type of situation is represented with only one exclusive gateway. Regarding the other exclusive gateway in the collaboration diagram, which merges the two possible branches, in the choreography diagram it is not reported since the developments of branches are not merging in a common situation obtainable with both branches. The decision is taken by the initiator of the last task before the gateway, i.e., Health Care Fund, which is in accordance with what is described in the collaboration.

Regarding the sequence of choreography tasks, we can see that they are linked by arrows defining the flow in which choreography tasks must be performed.

As anticipated in Section 4.1, data objects like "Pathology documentation" and "Available dates" are not reported in the choreography diagram, but in the next steps this kind of situation is faced.

The contract represented through the choreography diagram is composed of the same steps that need to be reached by participants in order to achieve a common business situation and the general sense has not changed from the collaboration diagram. The participant that started the flow in the collaboration diagram was Company Employee with the information sent to Health Fund Care; in the choreography diagram, the initiator of the first task is Company Employee that sends personal information to Health Fund Care. Even if we analyze the decision if the request is compliant or not, in the collaboration diagram the decision was taken by Health Fund Care, with Company Employee and Medical Office influenced by it; even in the choreography diagram, the decision and the

consequences involve the same participants. We can assert that the general sense has been conserved in the mapping.

Regarding the interactions in the choreography diagram, the sequence of actions is respected compared to the collaboration one: the order of initiators of tasks reflects the effective order of actions defined in the collaboration diagram, starting with Company Employee, passing through the decision taken by Health Care Fund and then ending with the Medical Office that sends the available dates for booking in case of accepted request.

Once we generated the choreography diagram, the next step is the extension of the diagram in order to include security properties. The steps to perform this kind of stage are detailed in Section 3.3, with an example of application visible in Figure 3.1.

5 | From Choreography to Corda

In this chapter we present the second transformation of our method, after we structured our choreography diagram and applied all the rules to manage security properties as discussed in Chapter 3 and 4, we need to generate our contract in Corda environment: the objects of contract are specified following precise rules for the mapping as described in Section 5.1, enriched by a series of algorithms that describe how to set names and attributes of objects on Corda contract defined in Section 5.2.

5.1. Conceptual mapping

Table 5.1 shows the conceptual mapping from choreography elements to Corda objects.

Choreography element	Corda object	Description
Choreography diagram	State, Contract	State object is mapped from the choreography diagram representing the evolution of the process represented in choreography diagram. Contract, which regulates the evolution of related state, is defined in the same moment.
Participants	Nodes of network	All participants in the choreography diagram become nodes in the private network created in Corda.
Choreography Task	Workflow, Command, Attribute of state	Choreography tasks represent interactions between two or more participants and Corda interactions among entities are represented by Workflows that generate transactions. Transactions consume states in input and generate output based on rules defined in the related command in contract. In case of special tasks added in the previous step of the model, it is expected an additional attribute to the state.
Connecting arrow	Input State	Connecting arrows indicate the succession of tasks in choreography diagram. In Corda, the concept is translated into the presence of input State in the transaction belonging to workflow generated by the next task.
Message	Attribute of state / Attachment	The nature of the message in the choreography diagram is based on the nature of the information sent.
Exclusive gateway	Attribute of state	Exclusive gateway entails a decision in choreography diagram, which is mapped into two attributes of the state that describe the possible options related to the decision and the validators of first category.

Table 5.1: Conceptual mapping

The first row of Table 5.1 describes the mapping between choreography diagram and the pair state-contract in Corda: as reported in 2.5, states represent the evolution of a fact and are shared among parties, while contracts regulate the behavior and validity of transactions involving specific types of states. This concept is matched in choreography diagram by analyzing the entire structure of the diagram, because it represents the evolution of the process composed of tasks, messages exchanged and gateways. Having this scenario, the mapping appears to be automatic.

The second row of Table 5.1 aims at defining participants of Corda network, by specifying identities of nodes: from choreography diagram they are visible to all participants to the business process, whether they are companies, public infrastructures or individuals. In this case, the natural mapping is to associate each of these participants to a node in the network.

The mapping described in the third row of Table 5.1 aims to represent the concept of actions that evolve the process: choreography tasks depict interactions between two or more participants and successive tasks create a flow that enables the process advancement, while workflows allow the creation of transactions responsible for the generation of new states. Furthermore, workflows contain transactions that are based on specific commands that regulate transactions defining rules and requirements during the execution, for instance regarding the attributes and the types of input and output states. According to this context, the generation of workflow and the corresponding command are mapped with choreography tasks.

The fourth row of Table 5.1 provides the concept of sequential execution of activities in the process: in choreography diagram, analyzing two successive tasks without the presence of other elements in between, a task cannot start until the previous task's execution is complete. This aspect can be translated in Corda by the presence of an input state in a transaction that is equal to the output state produced by the transaction coming from the previous task. The only exception is represented by the first task in the diagram, as it does not have any incoming arrows.

The fifth row of Table 5.1 is dedicated to the representation in Corda of messages sent and received in the choreography diagram. Based on the definition in BPMN modeling language, a message can represent a text message, numeric data or a file, thus the standard allows for a wide range of possibilities. According to it, the mapping in Corda foresees a double scenario: it can be done directly through an attribute of the State or as an attachment to the transaction generated in the workflow, following the rules specified in Section 5.2.

In the last row of Table 5.1 it is shown how to map the possibility of choice between two different paths, analyzing gateways in choreography diagram: based on the assumptions in Section 3.2, exclusive gateways are the type of gateway that we are interested in mapping in Corda, due to the presence of security property linked to the enforceability of decisions. In this case, it is sufficient to incorporate in the state a specific attribute that indicates all selectable paths after the gateway.

5.2. Rules of transformation

After we defined the mapping between concepts, it is needed a further step where we are going to schematize through algorithms the generation of contracts in Corda, based on the conceptual mapping described in Table 5.1 in Section 5.1.

In order to generate a deployable CorDapp, there is the necessity to integrate the output of algorithms with some activities, usually executed by programmers that are implementing the contract, indicated in the algorithm with (Exp) at the beginning of the line. For this reason, the main Algorithm 5.1 will recall the single algorithms and present further steps linked with additional activities to execute. The Algorithm 5.1 is structured in the following way: in line 2 the algorithm related to state and contract name definition is called to start defining the main objects of Corda; line 3 is the first complementary activity which is not regulated by an algorithm due to the fact that attributes of the state may be related to the context represented, for instance, the use of a unique identifier or the receiver attribute which may require a list or a single participant; lines 4-5 trigger algorithms for the generation of nodes in the network and for the definition of workflows and additional attributes of the state; line 6 refers to the initial phase in the workflow where the notary in the network is added for validation, in this case, there may be the presence of an additional notary or oracle required by a participant; lines 7-9 go into detail for what regards the input state, only if the workflow is not the first one, and in particular they refer to the search on the vault of the initiator to retrieve id of unconsumed transactions; line 10 deepens in the building of transaction, which requires to add all the components as possible attachments, commands, input and output state and collect signatures from participants which can require additional checks to the transaction; lines 11-12 activate the algorithms for commands and related rules definition; line 13 refers to the complementary rules to add to a specific command, for instance the rules related to specific attributes of the output state; line 14 triggers the algorithm for the representation of messages in Corda; lines 15-17 refer to the fact that if there is a message handled with attachment, there is needed a specific method for the upload and the sharing of the content in the network; line 18 is

linked to the algorithm that manages the presence of exclusive gateways.

Algorithm 5.1 Choreography to Corda

```

1: procedure CHOREOGRAPHY TO CORDA
2:   State and contract name definition: Algorithm 5.2
3:   (Exp) Define basic attributes of state
4:   Add nodes in Corda network: Algorithm 5.3
5:   Workflows and state attributes: Algorithm 5.4
6:   (Exp) Retrieve Notary information
7:   if not First Workflow then
8:     (Exp) Retrieve input state from vault
9:   end if
10:  (Exp) Complete transaction requirements
11:  Commands name definition: Algorithm 5.5
12:  Commands rules definition: Algorithm 5.6
13:  (Exp) Specify complementary rules
14:  Messages representation: Algorithm 5.7
15:  if Attachment in transaction then
16:    (Exp) Add specific methods for the upload of files
17:  end if
18:  Exclusive gateways representation: Algorithm 5.8
19: end procedure

```

The Algorithm 5.2 sets the name of Corda state and contract: after taking in input a choreography diagram in line 1, lines 3-4 describe the initialization of state and contract and in lines 5-7 the names of these two elements are generated directly from diagram name and then they are returned as the result of the algorithm.

The choreography diagram of our running example after the previous stage is depicted in Figure 3.2 and it represents the input of the Algorithm 5.2. Since the name of the choreography diagram is "Affiliated Visit", we will have AffiliatedVisitState and AffiliatedVisitContract as outputs.

Algorithm 5.2 State and contract name definition

```

1: input: Ch_diagram;
2: procedure STATE_AND_CONTRACT_NAMEDEFINITION
3:   State = NULL
4:   Contract = NULL
5:   replace NULL with Ch_diagramName + ' State' in State_Name
6:   replace NULL with Ch_diagramName + ' Contract' in Contract_Name
7:   return State, Contract
8: end procedure

```

Adding nodes to the Corda network by creating a list of all participants is performed through the Algorithm 5.3: as input in line 1, there will be all tasks in the choreography diagram and in line 3 the list of nodes is initialized as empty; in lines 4-11 the "for each" instruction aims to analyze each task of the tasks list and in lines 5-6 it is checked if the initiator of task is already present in the nodes list and, if negative, it is added to the list; it is made the applied the same condition to the receivers of task in lines 8-9. At line 12, the nodes list will represent the output and it will contain the nodes of the network alongside the Notary node contained by default as expressed in 2.5.

Applying the algorithm to Figure 3.2, all tasks are taken as input and, starting from the first one, initiators and receivers are added to the nodes list: at the end of the algorithm the nodes of Corda network will be Company Employee, Health Care Fund and Medical Office.

Algorithm 5.3 Adding nodes to the network

```

1: input: Ch_tasks_list;
2: procedure ADDING_NODES_IN_CORDA_NETWORK
3:   nodes_list  $\leftarrow$  {0}
4:   for each Ch_task  $\in$  Ch_tasks_list do
5:     if not Ch_taskInitiator  $\subseteq$  nodes_list then
6:       nodes_list  $\leftarrow$  Ch_taskInitiator
7:     end if
8:     if not Ch_taskReceivers  $\subseteq$  nodes_list then
9:       nodes_list  $\leftarrow$  Ch_taskReceivers
10:    end if
11:  end for
12:  return nodes_list
13: end procedure

```

The Algorithm 5.4 aims to create workflows, with related workflowInitiator as WfInit and workflowResponder as WfResp, and setting specific attributes in the state. In line 1, it takes all choreography tasks ordered and the state as input. Lines 3-4 are dedicated to initializing attributes for managing multiple gateways and data objects in the choreography diagram. Lines 5-7 represent the sentences used for the evaluation of tasks added in 3.3. Line 8 initializes workflows to empty. "For each" instruction between lines 9-38 holds all conditions evaluated to the generation of workflows by analyzing each task of the tasks list: lines 10-12 aim to set names of workflows; lines 13-15 set initiator and receivers of output state and adds it to the transaction output state; lines 16-20 evaluate if the task is the first one of the process and, if negative, it is set the transaction input state equals to the previous transaction output state related to the flow of the previous task, while if positive, the transaction input state is set to empty; line 21 adds the workflows set before to the workflows to generate; lines 22-28 evaluate if there are some tasks added for managing privacy or enforceability of decision properties and, based on the name of the task, it is added a different attribute in the state; lines 29-37 check if the task has the message from both initiator and receivers and, in positive case, it is set the second workflow related to the task, i.e. workflowReply with related workflowReplyInitiator and WorkflowResponder, by reversing initiator and receiver of the task and it is linked to the first flow created by the transaction input state; line 39 returns workflows and state.

With reference to the running example in Figure 3.2, the algorithm analyzes all choreography tasks present in the choreography diagram and then try to generate workflows and attribute for the state: starting from the first task, i.e., "New Request of affiliated visit", a new workflow is immediately created with the name "NewRequestOfAffiliated-VisitFlow" and, at the same time, the pair of workflows related to initiator and receiver, respectively "NewRequestOfAffiliatedVisitFlowInitiator" and "NewRequestOfAffiliated-VisitFlowResponder". Afterwards, parameters in the output state generated by the transaction in the workflow initiator are set: in this case, the initiator will be equal to Company Employee, while the receiver will be equal to Helth Care Fund. In this case, since the task is the first one of the choreography diagram, there is no necessity to add an input state.

Other conditions, in case of the first task, will be negative because it is not related to data sharing and it doesn't contain two messages. If we focus on the second task, we can deeply go inside the conditions related to the sharing of data among participants: the task named "Privity_Sharing Data" triggers the first condition that adds the attribute "sharingDataPrivityOne", set to false initially, to the state. In this example, the other conditions are not true because there is not a task dedicated to the sharing of data on which decision is taken, because the "Pathology documentation" is shared in the second

task and we are sure that all participants influenced by that decision know the data upon which it is taken. The other condition related to the presence of two messages in the task, one sent by the initiator and one by the receiver, in this example is not explored due to the configuration of the choreography diagram. These two conditions are necessary to take into consideration all the examples that refer to this type of case.

Algorithm 5.4 Workflows and state attributes

```

1: input: Ch_tasks_list, State;
2: procedure WORKFLOWS_AND_STATE_ATTRIBUTES
3:   ProgrNumberPrivity = 0
4:   ProgrNumberDecisions = 0
5:   Privity_task = ' Privity_sharingData'
6:   Decision_task = ' Decision_sharingData'
7:   Decision_taskValidation = ' Assessment'
8:   Workflows ← {0}
9:   for each Ch_task ∈ Ch_tasks_list do
10:     Wf_name = Ch_taskName + ' Flow'
11:     WfInit_name = Wf_Name + ' Initiator'
12:     WfResp_name = Wf_Name + ' Responder'
13:     WfInit_outputState_initiator = Ch_task_initiator
14:     WfInit_outputState_receivers = Ch_task_receivers
15:     WfInit_transactionOutputState ← WfInit_outputState
16:     if not Ch_task isFirstTaskOfProcess then
17:       WfInit_transactionInputState ←
18:         Previous_WfInit_transactionOutputState
19:     else
20:       WfInit_transactionInputState ← {0}
21:     end if
22:     Workflows ← Wf, WfInit, WfResp
23:     if Privity_task isSubstringOf Ch_taskName then
24:       State addAttribute State_sharingDataPrivity_ProgrNumberPrivity +
25:       1 = false
26:     else if Decision_task isSubstringOf Ch_taskName then
27:       State addAttribute State_sharingDataDecision_ProgrNumberDecisions +
28:       1 = false
29:     else if Decision_taskValidation isSubstringOf Ch_taskName
30:     then
31:       State addAttribute State_validatorsSecondCategory =
32:       false
33:     end if
34:     if Ch_task hasTwoMessages then
35:       WfReply_name = Ch_taskName + ' FlowReply'
36:       WfReplyInit_name = WfReply_name + ' Initiator'
37:       WfReplyResp_name = WfReply_name + ' Responder'
38:       WfReplyInit_transactionOutputState_initiator =
39:       Ch_task_receivers
40:       WfReplyInit_transactionOutputState_receivers =
41:       Ch_task_initiator
42:       WfReplyInit_transactionInputState =
43:       WfInit_transactionOutputState
44:       Workflows ← WfReply, WfReplyInit, WfReplyResp
45:     end if
46:   end for
47:   return Workflows, State
48: end procedure

```

The Algorithm described in 5.5 defines the commands' name of the contract: in line 1 the workflows list is taken as input; in line 3 the commands set is initialized to empty, during the algorithm it will be filled; the "For Each" instruction in lines 4-7 takes each workflow from workflows list and assigns the name to the related command by cutting the word "Flow" from the name of the workflow itself and it adds the command to the commands set; line 8 represents the output of the algorithm, with the commands set created.

Applying the algorithm to our example in Figure 3.2, workflows coming from choreography tasks have been selected, once per time, and a command is then generated with the same name of related workflow without the word "Flow". They are not taken directly from the choreography tasks because in the case of tasks with sending and receiving messages, two workflows are generated and the name would be repeated for both of them. For instance, the workflow related to the first task, i.e., "NewRequestOfAffiliatedVisitFlow", generates a command with the name "NewRequestOfAffiliatedVisit".

Algorithm 5.5 Commands definition

```

1: input: Workflows_list;
2: procedure COMMANDS_NAMEDEFINITION
3:   Commands  $\leftarrow$  {0}
4:   for each Workflow  $\in$  Workflows_list do
5:     Command_Name = Workflow_Name minus 'Flow'
6:     Commands  $\leftarrow$  Command
7:   end for
8:   return Commands
9: end procedure

```

In the Algorithm 5.6, we are going to describe how commands can specify some rules that transactions defined in the Algorithm 5.4 have to respect when they are executed: in line 1 the commands and transactions list are taken as input; the "For Each" instruction in lines 3-16 loops for each command in commands list and defines the rules of commands through several steps; in lines 4-6 sets for rules of that specific command are initialized to empty and it is selected the transaction from the workflows transactions list related to that command; lines 7-9 check if the transaction has an input state, i.e., it is the first one or not, and if the condition is verified then the rules related to the presence and the type of the input state are added; lines 10-12 explore the case of no input state present in the transaction, in this case, the rule added indicates that there won't be an input state for that transaction; lines 13-14 add the rules to the presence and the type of output state and we are sure that this rule will be present in each command since we defined in Section

3.2 that each transaction produces an output state; line 15 adds the rules generated to the command rules set of the command taken into consideration; line 17 refers to the output of the algorithm, i.e., the commands updated with the defined rules.

Adapting the algorithm to our example in Figure 3.2, we can notice that the first transaction is related to the workflow coming from the choreography task "New Request of affiliated visit": since this transaction is the first one, it doesn't have an input state, for this reason, the rule to add in this case is the one related to the no presence of an input state. For all the other transactions, as stated in Section 3.2, the rule to add is related to the presence of an input state. Regarding the output state, all transactions have to include that type of rule given that each transaction will generate an output state.

Algorithm 5.6 Commands rules definition

```

1: input: Commands_list, Workflows_transactions_list;
2: procedure COMMANDS_RULESDEFINITION
3:   for each Command  $\in$  Commands_list do
4:     Command_rules  $\leftarrow$  {0}
5:     Rules  $\leftarrow$  {0}
6:     Transaction = Workflow_transaction  $\in$  Workflows_transactions_list
       related to Command
7:     if Transaction hasInputState then
8:       Rules  $\leftarrow$  Transaction hasInputState
9:       Rules  $\leftarrow$  Transaction_inputStateType
10:    else
11:      Rules  $\leftarrow$  Transaction hasNoInputState
12:    end if
13:    Rules  $\leftarrow$  Transaction hasOutputState
14:    Rules  $\leftarrow$  Transaction_outputStateType
15:    Command_rules  $\leftarrow$  Rules
16:  end for
17:  return Commands
18: end procedure

```

The Algorithm 5.7 aims to describe how messages, based on their nature, are translated in Corda: in line 1 it takes as input the state, all choreography tasks and workflows related to tasks; the "For each" instruction between lines 3-18 contains different checks that are performed in order to decide how to map messages in Corda, analyzing each choreography task in the list; lines 4-5 describes the first actions when we analyze a message in a task,

thus the descriptive message of the transaction is set and the attribute related to the task is set to true in the output state of the transaction; lines 6-8 consider the option of having a non textual message to send and, in positive case, an attachment is added to the transaction and the content of the message is transferred as a file; lines 9-15 deepen the hypothesis that a task may have two messages, one sent by the initiator and one sent by receivers; if we are in that case, we are going to set the same attributes of lines 4-5 but in the workflowReplyInitiator; lines 12-14 check if the content of message sent by the receivers of task have to be transferred as a file with an attachment to the transaction in the workflowReplyInitiator; lines 17 represents the output of the algorithm described, the workflows with transactions updated.

Referring to our example in Figure 3.2, all tasks and workflows are taken into consideration and, by analyzing the "Privity_sharingData" tasks, their messages are mapped with an attachment to the transactions generated by respective flows, while messages of other tasks are depicted with an attribute inside the flow that changes value based on the message it is sending and they are going to set the attribute related to their change in the flow to true.

Algorithm 5.7 Messages representation in Corda

```

1: input: Ch_tasks_list, Workflows;
2: procedure MESSAGES_REPRESENTATION
3:   for each Ch_task ∈ Ch_tasks_list do
4:     WfInit_descriptiveMessage = Task_Message
5:     WfInit_transactionOutputState_contentOfMessage = true
6:     if not Ch_task_messageFromSender isTextual then
7:       WfInit_transactionAttachment = Task_Message
8:     end if
9:     if Ch_task hasTwoMessages then
10:      WfReplyInit_descriptiveMessage = Task_Message
11:      WfReplyInit_transactionOutputState_contentOfMessage = true
12:      if not Ch_task_messageFromReceivers isTextual then
13:        WfReplyInit_transactionAttachment = Task_Message
14:      end if
15:    end if
16:  end for
17:  return Workflows
18: end procedure

```

The Algorithm 5.8 defines the translation of exclusive gateways present in choreography diagram into Corda: in line 1 the state and all exclusive gateways from the diagram are taken as input; lines 3-8 specify for each exclusive gateway what are the actions to be performed; lines 4-6 examine all the possible branches for a specific gateway and sets the attribute related to each of possible branch to false, indicating that no decision has been taken because no branches are selected; line 7 adds an attribute related to the validators of first category set to false, because we are sure that at least validators of first category will be present; line 9 returns the state with all changes made in the previous lines.

In our example in Figure 3.2, it is taken the only exclusive gateway present and the `AffiliatedVisitState` generated before; then it is taken into examination the gateway and the possible branches, i.e., acceptance or refusal, and attributes accepted and refused are added to the state and set to false. After that, the attribute related to validators of first category, with a false value, is added to the state.

Algorithm 5.8 Exclusive gateways in Corda

```

1: input: Ch_exclusive_gatewaylist, State;
2: procedure EXCLUSIVE_GATEWAYS_REPRESENTATION
3:   for each Ch_exclusive_gateway  $\in$  Ch_exclusive_gateway_list do
4:     for each Ch_exclusive_gateway_branch  $\in$  Ch_exclusive_gateway do
5:       State_decisionBranch = false
6:     end for
7:     State addAttribute State_validatorsFirstCategory = false
8:   end for
9:   return State
10: end procedure

```

All the algorithms described until now have to be contained in another algorithm that defines the order in which they are activated to automate some parts of the realization of `CorDapp`.

6 | Validation

This chapter shows the validation of the method proposed in this thesis. In order to carry out the validation of the method, we have implemented a software that automates part of the process. The main purpose of this phase is to check the correctness of the method through the analysis with real cases of different complexity. The validation phase is composed of three sections: the first one, Section 6.1, aims to validate the mapping and transformation from collaboration to choreography diagram extended with security properties; the second one, Section 6.2, analyzes the mapping and transformation from choreography to Corda in order to generate the skeleton of CorDapp through the application of algorithms described in Chapter 5; the last one, Section 6.3, contains the discussion on the results of two sections. Combining the results obtained in two sections we are going to validate the whole method.

6.1. Collaboration to Choreography

This section validates the mapping and transformation from collaboration to choreography diagram extended with security notations according to the rules expressed in Chapters 3 and 4.

The software aims to apply the theoretical mapping by generating an XML document¹ that represents the choreography diagram with security properties enforced through successive steps:

1. It takes as input an XML document, that represents the collaboration diagram modeled with SecBPMN2BC, containing the messages exchanged and participants' identities. Since the method doesn't focus on internal activities, they are not reported in the initial XML document as they wouldn't have been analyzed. This first step generates automatically the choreography tasks, represented by an XML document, that are contained in the choreography diagram.

¹XML document: "eXtensible Markup Language" document, a text file containing structured data. XML allows you to define tags and structure rules adaptable to specific domain needs. It is composed of hierarchical elements enclosed in tags, with the option to include attributes for additional information.

2. The XML generated is then enriched with elements linked to security properties such as exclusive gateways and data objects, which are inserted with specific tags. This activity is performed manually since the representation of security properties in the choreography diagram doesn't follow a 1:1 mapping. Instead, they are adapted to a new context following the guidelines described in Chapter 4. It may happen that internal tasks of the activities related to data objects or gateways are not reported in the choreography diagram: in this case, in order to include the security elements, there is the necessity to find a way to include them in the diagram. For these reasons, we preferred performing this activity manually. We obtain an XML document that represents tasks and security elements of the choreography diagram enriched with security properties.
3. The third step takes as input the XML document of the previous phase and then carries out the enforcement of the security properties generating the additional choreography tasks as described in Section 3.3. These new tasks, with to the previous ones, represent the final choreography diagram that is described by an XML document. To represent the sequence of tasks, start and end events, specific tags of the XML document related to incoming and outgoing links are used: if the tag `<incoming>` is not present, the task is the first one and it means that the predecessor is the start event; if only the `<incoming>` tag is present, it means that after it, there is an end event; otherwise, in case of both tags present, the task is connected to a predecessor and a successor. This step marks the end of the first phase of the validation.

What we expect is that the output of the process, i.e., the choreography diagram, respects the correctness of the process in the following aspects: participants included in the diagram; interactions represented, the order in which they occur and the correct participants involved; general sense of the contract respected; inclusion of objects with reference to security properties such as exclusive gateways and data objects.

Realistic examples have been chosen taking into consideration the context represented and the high complexity. For this reason, collaboration diagrams present additional symbols that go even beyond our context. Despite this, we will focus on the security properties of privacy and enforceability of decisions.

6.1.1. Realistic examples

Birth certificate issue

The first example describes the registration of a birth certificate: there are two pools in the collaboration diagram, i.e., citizen and the citizen registry birth certificate, which are

the participants. The flow starts with the citizen who scans the ID document and fills out the related application form to register the birth certificate. Since this module has been completed and sent to the citizen registry birth certificate, the information from the application form are collected and then the birth is registered. At the end, the birth certificate is sent to the citizen who can print it.

In this example, there are no gateways, because there are no decisions to take due to the linear process which is composed by a defined sequence of activities. For what regards the presence of data objects linked with the security property of privity, we can report three data objects: the ID of citizen in electronic form related to a static level of privity; the birth application form, again related to a static level of privity, and the birth certificate, linked to a private level of privity.

Applying the first step of mapping, the software produced the XML document visible in Listing 6.1, which depicts the choreography tasks of the choreography diagram in Figure 6.1.

```
<?xml version="1.0" encoding="UTF-8" ?>
<choreographydiagram name="BirthCertificateIssue">
  <choreographytasks>
    <choreographytask name="Request of new birth certificate">
      <message>Request</message>
      <initiator>1</initiator>
      <receivers>2</receivers>
    </choreographytask>
    <choreographytask name="Birth certificate registered">
      <message>Notification</message>
      <initiator>2</initiator>
      <receivers>1</receivers>
    </choreographytask>
  </choreographytasks>
  <participants>
    <participant id="1">Citizen</participant>
    <participant id="2">Citizen birth certificate issue</participant>
  </participants>
</choreographydiagram>
```

Listing 6.1: XML document - Birth Certificate Issue

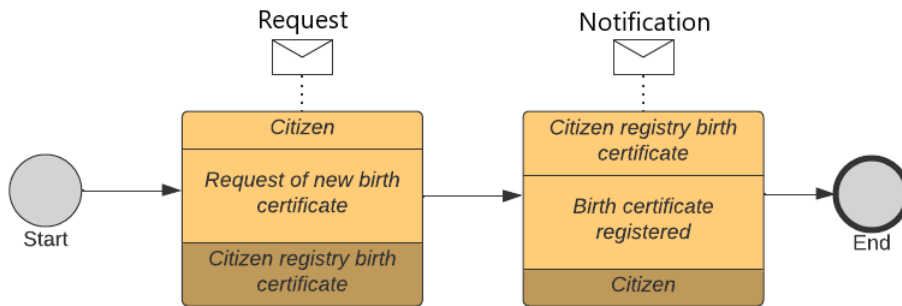


Figure 6.1: Choreography diagram of birth certificate issue

In the second step, the choreography diagram in Figure 6.1 is enriched manually with security notations and the result is reported in Figure 6.2. The XML document of the diagram is showed in Listing 6.2

```
<?xml version="1.0" encoding="UTF-8" ?>
<choreographydiagram name="BirthCertificateIssue">
  <choreographytasks>
    <choreographytask name="Request of new birth certificate">
      <message> Request </message>
      <dataobject privity_level="static" owner="1" others="2">Id
        document</dataobject>
      <dataobject privity_level="static" owner="1" others="2">Application
        form</dataobject>
      <initiator> 1 </initiator>
      <receivers> 2 </receivers>
    </choreographytask>
    <choreographytask name="Birth certificate registered">
      <message> Notification </message>
      <dataobject privity_level="private" owner="2" others="1" >Birth
        certificate</dataobject>
      <initiator> 2 </initiator>
      <receivers> 1 </receivers>
    </choreographytask>
  </choreographytasks>
  <participants>
    <participant id="1"> Citizen </participant>
    <participant id="2"> Citizen birth certificate issue </participant>
  </participants>
</choreographydiagram>
```

Listing 6.2: XML document enriched with security properties - Birth Certificate Issue

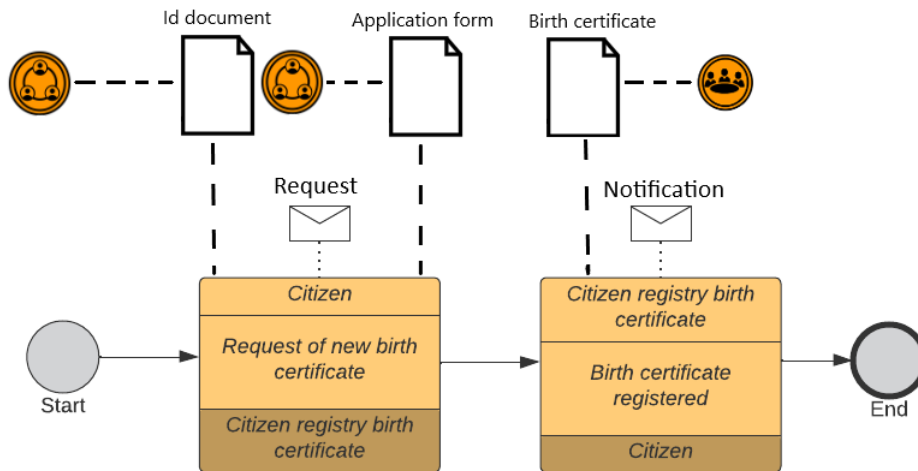


Figure 6.2: Choreography diagram extended with security notations of birth certificate issue

Subsequently, the security properties are enforced and the additional tasks generated automatically are reported in Listing 6.3.

```
<?xml version="1.0" encoding="UTF-8" ?>
<choreographydiagram name="BirthCertificateIssue">
  <choreographytasks>
    <choreographytask name="Privity_SharingData">
      <message>Id document</message>
      <initiator>1</initiator>
      <receivers>2</receivers>
    </choreographytask>
    <choreographytask name="Privity_SharingData">
      <message>Application form</message>
      <initiator>1</initiator>
      <receivers>2</receivers>
    </choreographytask>
    <choreographytask name="Privity_SharingData">
      <message>Birth certificate</message>
      <initiator>2</initiator>
      <receivers>1</receivers>
    </choreographytask>
  </choreographytasks>
</choreographydiagram>
```

```

<participants>
  <participant id="1">Citizen</participant>
  <participant id="2">Citizen birth certificate issue</participant>
</participants>
</choreographydiagram>

```

Listing 6.3: XML document of additional tasks - Birth Certificate Issue

The final choreography diagram is then obtained by combining the generated additional tasks with the others already present, providing them a sequential order as described in Section 2.5: the additional tasks are included and the security properties are enforced as reported in the choreography diagram in Figure 6.3, with the related Listing 6.4.

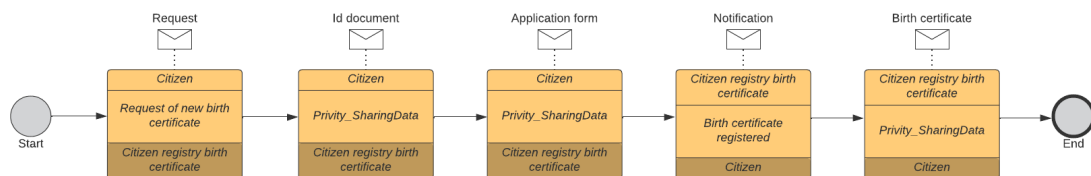


Figure 6.3: Choreography diagram from the extended choreography diagram of birth certificate issue

```

<?xml version="1.0" encoding="UTF-8" ?>
<choreographydiagram name = "BirthCertificateIssue">
  <choreographytasks>
    <choreographytask name="Request of new birth certificate" seq="task_1">
      <message>Request</message>
      <initiator> 1 </initiator>
      <receivers> 2 </receivers>
      <outgoing> task_2 </outgoing>
    </choreographytask>
    <choreographytask name="Privity_SharingData" seq="task_2">
      <message>Id document</message>
      <initiator>1</initiator>

```

```

    <receivers>2</receivers>
    <incoming> task_1 </incoming>
    <outgoing> task_3 </outgoing>
</choreographytask>
<choreographytask name="Privity_SharingData" seq="task_3">
    <message>Application form</message>
    <initiator>1</initiator>
    <receivers>2</receivers>
    <incoming> task_2 </incoming>
    <outgoing> task_4 </outgoing>
</choreographytask>
<choreographytask name="Birth certificate registered" seq="task_4">
    <message>Notification</message>
    <initiator> 2 </initiator>
    <receivers> 1 </receivers>
    <incoming> task_3 </incoming>
    <outgoing> task_5</outgoing>
</choreographytask>
<choreographytask name="Privity_SharingData" seq="task_5">
    <message>Birth certificate</message>
    <initiator>2</initiator>
    <receivers>1</receivers>
    <incoming> task_4 </incoming>
</choreographytask>
</choreographytasks>
<participants>
    <participant id="1"> Citizen </participant>
    <participant id="2"> Citizen birth certificate issue </participant>
</participants>
</choreographydiagram>

```

Listing 6.4: XML document of final choreography diagram - Birth Certificate Issue

Hospital televisit external

The second example describes an hospital televisit requested by a pediatric patient, authorized by the family of patient, organized by the hospital staff and then executed by an external specialised group of professionals. It is composed of four pools, one for each participant: pediatric patient, hospital staff, family and specialised group.

The flow starts from the pediatric patient who requests an appointment on a set date to

the hospital staff who, based on the patient's situation, must ask the patient's family for the treatment of medical information related to her. After the authorization to proceed, it is requested the teleconsultation of the specialised group that executes the televisit to the patient and generates the medical report after the discussion with hospital staff. In the end, the final report is then sent to the patient.

Regarding the exclusive gateways marked with the security property of enforceability of decisions, the first gateway is related to the consent for processing medical information, which is linked to the security property with a level set to private. The second one is related to the availability of the external specialized group, which is linked to the security property with a level set to public.

Analyzing data objects marked with the security property of privity, we can notice that there are four data objects to consider: the first one regards the purpose for content that is linked to the public level of privity; the second one is the signed content related to the private level of privity; then, the patient record and the specialistic report which are related respectively to the static level and to the private level of privity.

The output obtained by the software is the choreography diagram reported in Figure 6.4: due to the dimensions of the diagram, for a better understanding all diagrams of this example are reported at Hospital Televisit External².

²https://github.com/alessandrodirenzo/Thesis_DiRenzo_Corda_project/tree/main/Diagrams%20images/HospitalTelevisitExternal

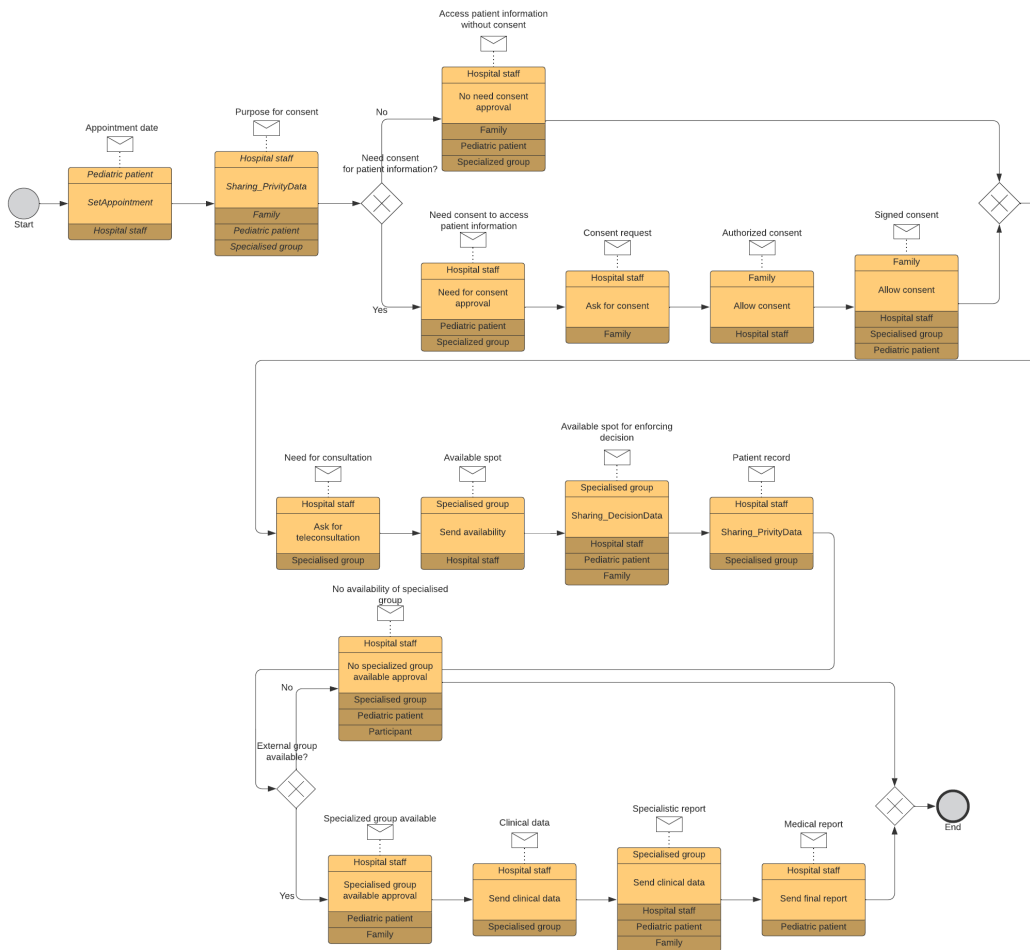


Figure 6.4: Hospital teleconsultation external choreography diagram

Hospital teleconsultation The third example describes an hospital teleconsultation having as participants family, doctor and pediatric patient. There are two starting events as visible in pool of family and pediatric patient: the sequence after the events are regulated by the sequence of interactions with the doctor, which enters the diagram after the request of the family about the availability for a teleconsultation. If there are free slots, the doctor sends the communication about the date chosen and waits for the consent notification from the pediatric patient, which has already produced it and now sent it to the doctor. The consent is then checked and, based on the response, the teleconsultation can happen with questions from the doctor’s side and answers about the symptoms from the pediatric patient’s side. As the last activity, the doctor checks the pediatric patient’s

record and updates the final report read by the pediatric patient.

Regarding exclusive gateways marked with the security property of enforceability of decisions, in this example, we find only an exclusive gateway related to the decision taken by the doctor about the consent sent by the pediatric patient, enforced with a user defined level.

Data objects linked to the security property of privity are three: the consent form signed by the pediatric patient, the patient health record consulted by the doctor and the final report stipulated by the doctor and accessible to the pediatric patient, all linked to a private level of privity.

The output of the software is the choreography diagram shown in Figure 6.5: due to the dimensions of the diagram, for a better understanding all diagrams of this example are reported at Hospital teleconsultation³.

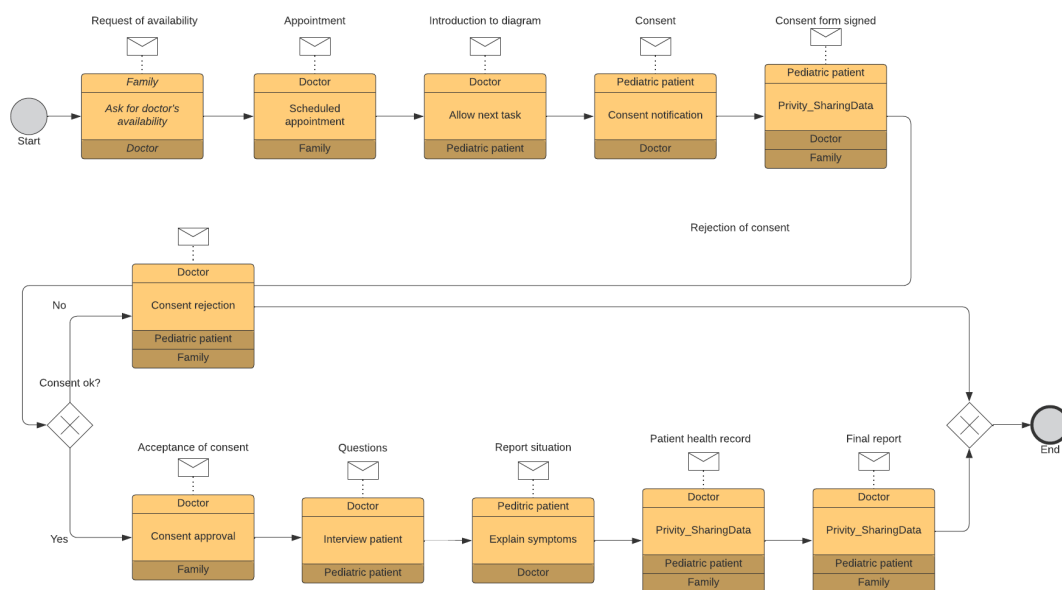


Figure 6.5: Hospital teleconsultation choreography diagram

6.1.2. Corner cases

Tests executed have highlighted specific situations where the method produced an output not in accordance with the contract described in the initial form and it needed adaptations

³https://github.com/alessandroDIRENZA/Thesis_DiRenzo_Corda_project/tree/main/Diagrams%20images/HospitalTeleconsultation

to be compliant. The corner cases encountered are described below.

Corner case 1

The first situation to be discussed is when we are going to add security properties to the choreography diagram and there is a lower number of choreography tasks with respect to the data objects marked with security property of privity. In this case, the solution adopted is the following: based on the sequence in which data objects are represented in the collaboration diagram and the interactions in which they have to be accessed, data objects still not associated are added to a choreography task that already presents another data object. Multiple data objects associated with a task are characterized by the following rules: the order of access to data objects is from left to right, thus a data object in the left corner is shared among participants before the one in the right corner. This solution is accepted based on what is described in Section 4.1, where it is claimed the importance of respecting the sequence of interactions and the availability of the necessary data to address it. Furthermore, in Section 3.3, it is reported that data objects linked to the security property of privity have to be present in the choreography diagram to be managed with special tasks. This corner case highlights the fact that this step is not automatic and several adaptations might be necessary.

Corner case 2

In the third case analyzed, a corner case is represented by the start event in the pool of the pediatric patient. That event is not related to a message coming from another participant, but the pediatric patient is the one who sends the message with the signed consent. The software produced as output the tasks contained into the choreography diagram depicted in Figure 6.6.

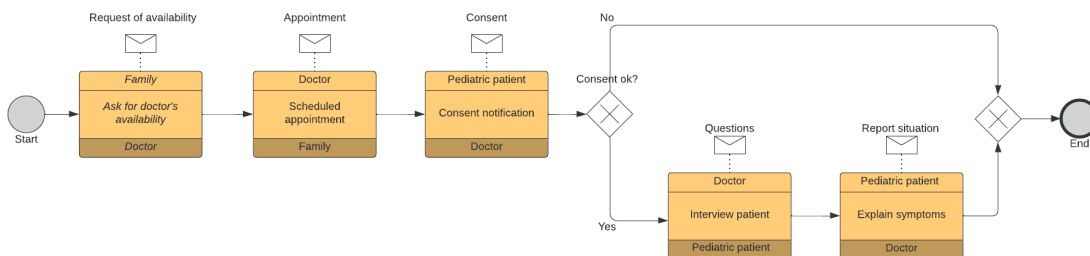


Figure 6.6: Hospital teleconsultation choreography diagram not acceptable

As reported in Figure 6.6, the choreography diagram is not valid due to the presence of a task where an initiator is not present in the tasks before, i.e., the pediatric patient

in the task "Consent notification". In this case, the solution is to add an interaction represented by a task where the initiator of the wrong task is present as receiver and the initiator of the previous task is again the initiator of this additional task. In this case, the additional task presents the doctor as initiator and as receiver the pediatric patient in a task called "Allow next task". The message sent by the doctor to the pediatric patient is symbolic, since the only purpose of this task is to allow the next interaction. This type of solution modifies the flow adding a task not present in the initial collaboration diagram. Despite that, one of the guidelines to follow for the transformation from collaboration to choreography is to preserve the general sense of the contract: in this case, the only way to allow the interaction, without acting directly on it, is to include the initiator in the previous task. Other interactions are not modified. In this case, a manual activity has been necessary to modify the output of the software to make the choreography diagram acceptable. The new XML document is then taken as input for the next step.

Corner case 3

Realistic cases of "Hospital televisit external" and "Hospital teleconsultation" present a data object related to the security property of privacy, which is not introduced by a specific participant, thus we cannot claim who is responsible for the introduction of the document. Moreover, there are participants who access that data object. In this situation, the participant who accesses first the data object is the one who shares it with others as indicated by the level of privacy. In the example "Hospital televisit external", data object "Patient record", accessed first by the hospital staff and then by the specialized group, is shared in a special task by the hospital staff after the "Ask for consultation" task because it has to be accessed by the specialised group in the next tasks. In the example "Hospital teleconsultation", data object "Patient final record" is accessed by the doctor and, in this case, it is shared after the last task by the doctor based on the level of privacy related to data objects. Since we are not assuming that some data can be stored in a common database accessible by the participants, this solution allows to include all data objects associated with the security property of privacy in the choreography diagram as specified in Section 3.2.

6.1.3. Results

Birth certificate issue

In the first example, analyzing what the software produced as output, starting from the Listing 6.1 and the associated Figure 6.1, the two interactions in the collaboration diagram have been represented correctly with the presence of both participants. In this example, the difference in what we are focusing on in the collaboration rather than the choreography

diagrams is shown: all the activities performed by citizen registry birth certificate from the "Process birth application form" to the "Cross check information on birth certificate" are not reported in the choreography diagram. They are considered as started after the reception of the message in the "Request of new birth certificate" task and executed when the "Birth certificate registered" task is initiated. We can ensure that the sequence of interactions between participants is respected. Regarding data objects, as discussed in Section 6.1.2, the situation represents a corner case handled in Figure 6.2. All data objects linked to the privacy security property present in the collaboration diagram are represented in the choreography diagram. The additional tasks generated automatically for the enforcement of security properties are shown in Listing 6.3 and then included in the choreography diagram as represented in Figure 6.3. Data objects are shared in the dedicated tasks based on the level of privacy associated with single data. We can mark as positive the first phase of mapping from collaboration to choreography diagram for this example.

Hospital televisit external

The second example shows how the software produced an output that is characterized by an elevated number of choreography tasks due to the higher number of interactions with respect to other examples. This example represents a case where the intermediate end events in each pool are not reported in the choreography diagram, as depicted in Section 4.1: the exclusive gateway "Available" in the pool of specialised group is characterized by an intermediate end event if the group is not available, but this is encapsulated in the choreography diagram in the negative path after the available slots sent by the specialised group. All the interactions in the collaboration diagram are included with the right participants. Regarding data objects, all of them have a task dedicated to the sharing of data based on the level of privacy associated with each data object. Data object "Patient record" has been managed as a corner case detailed in Section 6.1.2. The enforceability of decisions related to the exclusive gateways in the choreography diagram has been managed by the software: for both exclusive gateways, the software generated an additional task for the validators of the second category. In order to validate the decision taken in the exclusive gateway related to the availability of specialised group, an additional task for the sharing of data related to the decision has been generated by the software in order to allow the validation. We can therefore confirm that the results obtained in this phase of the method are in agreement with what we expected.

Hospital teleconsultation

Regarding the third example, the software produced an output not acceptable based on the choreography diagram rules: as described in Section 6.1.2, this situation represents a

corner case. Adopted the new choreography diagram after the modification, the software produced valid outputs in the next steps. We can observe that the sequence of the interactions is respected, even with the additional task with the doctor and the patient. Regarding data objects, the "Patient health record" highlights a corner case detailed in Section 6.1.2. All data objects have the task for the sharing and the order of access is respected. Analyzing exclusive gateways, the only one associated with the enforceability of decisions is related to consent and it is present in the choreography diagram. The level of enforceability is user-defined, thus the definition of the set of participants that validate the decision is left to the modeler of the diagram: in this case, it has been chosen to validate the decision with both patient and family to show the most complex case of validation. In case of a negative answer, there are only validators of second category since there are no interactions in that branch. On the other hand, in case of acceptance of consent, there are both validators of the first and second categories in successive tasks. The general sense of the contract has not been modified, all the participants are included and, with the adaptation related to corner cases, the results of this phase are accepted.

6.2. Choreography to Corda

This section aims to validate the mapping and transformation from the choreography diagram produced as output of the first mapping described in Section 6.1. Algorithms detailed in Section 5.2 have been implemented in order to transform the choreography diagram into a Corda smart contract. In this second phase of validation, the software takes as input the XML of the choreography diagram generated in the previous phase and generates the Java classes that represent the skeleton of the CorDapp. The generated classes include a state with attributes that evolves over time, a contract that regulates its evolution and workflows that create transactions to consume and generate new states. What we expect from this part of validation is that the skeleton of CorDapp is generated including the following elements: nodes representing participants; state and contract; workflows and transactions with basic rules of commands defined.

6.2.1. Algorithms application

Birth certificate issue

Applying the algorithms to the choreography diagram of the first example, we obtained the state "BirthCertificateIssue" and the contract "BirthCertificateIssueContract". For each choreography task, there is a pair of workflows initiator and responder. Related to the transaction in workflows there is a command in contract that regulates the pres-

ence of input state and the value of the attribute message which can be empty or not: empty if it is a workflow that shares files, otherwise not empty. Since there are no exclusive gateways, the attributes related to possible branches and to validators of the first and second categories are not present. Data objects are shared in the transactions coming from "PrivitySharingDataOneFlow", "PrivitySharingDataTwoFlow" and "PrivitySharingDataThreeFlow" workflows. In this case, the additional attributes for the state regard only the two workflows not related to the sharing of data. Java class of the state "BirthCertificateIssue" and "BirthCertificateIssueContract" is described in Listing 6.5.

```
public class BirthCertificateIssue implements ContractState{

    UniqueIdentifier idState;
    String message;
    Party initiator;
    List<Party> receivers;
    boolean datashared1;
    boolean datashared2;
    boolean datashared3;

    public BirthCertificateIssue(UniqueIdentifier idState, String
        message, Party initiator, List<Party> receivers, boolean
        datashared1, boolean datashared2, boolean datashared3){
        this.idState = idState;
        this.message = message;
        this.initiator = initiator;
        this.receivers = receivers;
        this.datashared1 = datashared1;
        this.datashared2 = datashared2;
        this.datashared3 = datashared3;
    }

    public UniqueIdentifier getidState() {
        return idState;
    }

    public String getmessage() {
        return message;
    }
}
```




```
public Party getinitiator() {  
    return initiator;  
}  
  
public List<Party> getreceivers() {  
    return receivers;  
}  
  
public boolean isdatashared1() {  
    return datashared1;  
}  
  
public boolean isdatashared2() {  
    return datashared2;  
}  
  
public boolean isdatashared3() {  
    return datashared3;  
}  
  
}
```

Listing 6.5: State class - Birth Certificate Issue

After the generation of classes and participants, nodes are deployed: Figure 6.7 shows nodes of citizen, registry birth certificate and default notary deployed in Corda environment.

```


Ah, Friday.
My second favourite F-word.

--- Corda Community Edition 4.9 (265389c) -----


Logs can be found in           : /home/salnitri/cordaDeployment/cordapp-template-java/build/nodes/Notary/logs
! ATTENTION: This node is running in development mode! This is not safe for production deployment.
Advertised P2P messaging addresses : localhost:10002
RPC connection address           : localhost:10003
RPC admin connection address     : localhost:10043
Loaded 2 CorDapp(s)              : Contract CorDapp: Template Contracts version 1 by vendor Corda Open Source with 1
licence Apache License, Version 2.0, Workflow CorDapp: Template Flows version 1 by vendor Corda Open Source with licence Apa
che License, Version 2.0
Node for "Notary" started up and registered in 23.75 sec

Welcome to the Corda interactive shell.
You can see the available commands by typing 'help'.

Fri Sep 15 14:56:54 UTC 2023>>> Running P2PMessaging loop

```

```


Ah, Friday.
My second favourite F-word.

--- Corda Community Edition 4.9 (265389c) -----

Logs can be found in           : /home/salnitri/cordaDeployment/cordapp-template-java/build/nodes/Citizen/logs
! ATTENTION: This node is running in development mode! This is not safe for production deployment.
Advertised P2P messaging addresses : localhost:10005
RPC connection address           : localhost:10006
RPC admin connection address     : localhost:10046
Loaded 2 CorDapp(s)              : Contract CorDapp: Template Contracts version 1 by vendor Corda Open Source with 1
licence Apache License, Version 2.0, Workflow CorDapp: Template Flows version 1 by vendor Corda Open Source with licence Apa
che License, Version 2.0
Node for "Citizen" started up and registered in 23.42 sec

Welcome to the Corda interactive shell.
You can see the available commands by typing 'help'.

Fri Sep 15 14:55:38 UTC 2023>>> Running P2PMessaging loop

```

```


I got a universal remote control yesterday.
I thought, this changes everything.

--- Corda Community Edition 4.9 (265389c) -----

Logs can be found in           : /home/salnitri/cordaDeployment/cordapp-template-java/build/nodes/Citizenregistryb
irthcertificate/logs
! ATTENTION: This node is running in development mode! This is not safe for production deployment.
Advertised P2P messaging addresses : localhost:10007
RPC connection address           : localhost:10008
RPC admin connection address     : localhost:10045
Loaded 2 CorDapp(s)              : Contract CorDapp: Template Contracts version 1 by vendor Corda Open Source with 1
licence Apache License, Version 2.0, Workflow CorDapp: Template Flows version 1 by vendor Corda Open Source with licence Apa
che License, Version 2.0
Node for "Citizen registry birth certificate" started up and registered in 24.25 sec

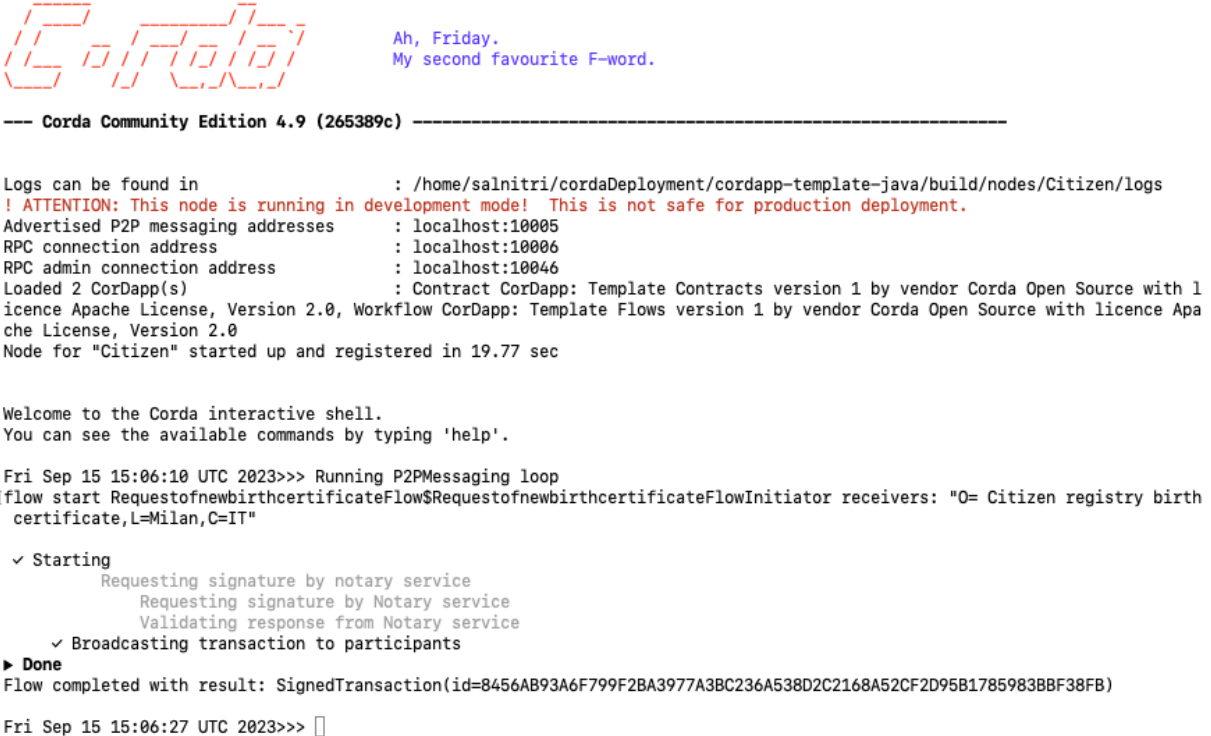
Welcome to the Corda interactive shell.
You can see the available commands by typing 'help'.

Fri Sep 15 14:57:20 UTC 2023>>> Running P2PMessaging loop


```

Figure 6.7: Nodes deployed: citizen, citizen registry birth certificate and notary

The transaction related to the first workflow "Request of new birth certificate" initiated by citizen and received by citizen registry birth certificate, is reported in Figure 6.8.



```


Ah, Friday.
My second favourite F-word.

--- Corda Community Edition 4.9 (265389c) ---

Logs can be found in           : /home/salnitri/cordaDeployment/cordapp-template-java/build/nodes/Citizen/logs
! ATTENTION: This node is running in development mode! This is not safe for production deployment.
Advertised P2P messaging addresses : localhost:10005
RPC connection address           : localhost:10006
RPC admin connection address     : localhost:10046
Loaded 2 CorDapp(s)             : Contract CorDapp: Template Contracts version 1 by vendor Corda Open Source with licence Apache License, Version 2.0, Workflow CorDapp: Template Flows version 1 by vendor Corda Open Source with licence Apache License, Version 2.0
Node for "Citizen" started up and registered in 19.77 sec

Welcome to the Corda interactive shell.
You can see the available commands by typing 'help'.

Fri Sep 15 15:06:10 UTC 2023>>> Running P2PMessaging loop
[flow start RequestofnewbirthcertificateFlow$RequestofnewbirthcertificateFlowInitiator receivers: "O= Citizen registry birth certificate,L=Milan,C=IT"

  ✓ Starting
    Requesting signature by notary service
    Requesting signature by Notary service
    Validating response from Notary service
  ✓ Broadcasting transaction to participants
▶ Done
Flow completed with result: SignedTransaction(id=8456AB93A6F799F2BA3977A3BC236A538D2C2168A52CF2D95B1785983BBF38FB)

Fri Sep 15 15:06:27 UTC 2023>>> []

```

Figure 6.8: Transaction generated by "Request of new birth certificate" workflow

In order to validate that the transaction signed by participants has been correctly stored inside personal vaults, we need to query the related vaults, as shown in Figure 6.9. As visible, the transaction is contained in both participants in row "ref:txHash", who now share the same state. For completeness, the screens are reported at corDapp screens⁴.

⁴https://github.com/alessandrodirenzo/Thesis_DiRenzo_Corda_project/tree/main/Diagrams%20images/CorDapp%20screens



(a) Citizen's vault

(b) Citizen registry birth certificate's vault

Figure 6.9: Vaults of participants

Hospital televisit external

The state and the contract coming from the application of the algorithms are respectively: "HospitalTelevisitExternal" and "HospitalTelevisitExternalContract". Since there are two exclusive gateways, each of them adds the attributes related to the validators of first and second category for the correspondent decision. In this case there is even the workflow for the sharing of data regarding the second gateway, with "DecisionSharingData". Data objects are shared in dedicated workflows with names "PrivitySharingData". In this case, it is visible that commands and workflows are increased in number and, at the same time, the additional attributes required for other workflows that are not involved in validation or sharing data are higher. All participants are represented with a node, in addition to the default node for the Notary.

Hospital teleconsultation

In the third example, the algorithms produced as output the state "HospitalTeleconsultation" and the related contract "HospitalTeleconsultationContract". For each choreography task, there are workflows of initiator and responder, with related transactions regulated by the commands sections in the contract. Commands present basic rules re-

lated to the presence of the input state and whether the attribute of message is empty or not. Regarding the decision related to the consent, there are attributes related to validators of first and second categories in the class of the state. Data objects are shared among participants in the workflows related to tasks with "PrivitySharingData" name through the method for the uploading of files. All participants are represented with a node, in addition to the default node for the Notary.

6.2.2. Comments

Analyzing the results of the algorithms applied to the realistic examples, there are some behaviors to discuss in detail. Contexts like "Hospital external televisit" highlight how the number of attributes associated with the state produced by the transactions may increase due to the number of tasks in the choreography diagram. Regarding the participants to the network, in the "Hospital external televisit" example there is the family who is a participant whose presence in the transactions is not guaranteed, but it depends on the evolution of the contract. The participant is, however, added to the private network from the algorithm at the beginning, not in the evolution of the contract by triggering a particular event. This situation can represent a corner case if some information are shared among all participants in a moment where that participant shouldn't be in the network. A similar case can be found even in the case of "Hospital teleconsultation" with specialised group, which can be not available but involved in the sharing of data. The output states added to the transaction in each workflow are composed of attributes set to the default value of false: in order to obtain a real evolution of the state, they must be set correctly. In order to specify a flow and the correct dependencies among transactions, in each commands section there are needed rules for specifying the values of attributes in each output state.

6.3. Discussion

Combining the results from both Section 6.1 and Section 6.2, the validation phase is completed. Examining the XML documents, the diagrams and the skeleton of CorDapps generated, the contracts follow the requirements of the contexts represented, with security properties enforced as detailed in the previous chapters. In order to replicate the experiments, the steps performed in both phases with related results are reported a repository Github⁵.

Despite the positive general result, there are some considerations about the method pro-

⁵https://github.com/alessandrodirenzo/Thesis_DiRenzo_Corda_project

posed:

- The mapping from collaboration to choreography diagram is the most critical phase: since there is almost a direct correspondence in the transformation from choreography diagram to Corda that doesn't allow to adapt to particular situations applying the algorithms, errors in the generation of the choreography diagram are propagated into the final CorDapp. For instance, in the case of "Hospital teleconsultation", the diagram generated at the beginning was not valid due to the presence of multiple start events that introduced a new participant as initiator without being involved in the previous tasks. In this case, without the correct changes, the algorithms would have generated a contract that doesn't respect the initial situation.
- The number of messages, the presence of exclusive gateways and data objects influence the number of tasks in the choreography diagram. As visible in the three examples, the number of tasks can vary significantly and it influences the number of attributes of the state after the transformation in a CorDapp. In a contract with a lot of interactions, exclusive gateways and data objects, this could represent a limitation for the scalability of the method that may lead to different choices in order to represent the different steps of the state.
- The software is related to the specific input where the necessary information are reported: if we don't specify exactly the participants who require the data in case of sharing of data objects or who are the validators of second category, the software is not able to retrieve the information analyzing the base version of the XML documents.
- In order to deploy a complete CorDapp, that is able to execute transactions following the rules specified in the commands section, there are needed specific imports and methods in addition to the skeleton generated by the software. Additional rules in the commands section need to specify the values of attributes related to the output state generated by transactions in order to evolve correctly.

7 | Conclusions

This thesis discusses a method for the enforcement of security properties of smart contracts for modeling processes such as confidentiality of data, i.e., privacy, and enforceability of decisions.

The method is composed of three parts: the contract described with SecBPMN2BC modeling language that is mapped into a BPMN Choreography with security notations; the extension of BPMN Choreography in order to include the security properties; the mapping and transformation of choreography diagram into a contract in Corda environment. In particular, the first mapping determines how to transform a collaboration into a choreography diagram based on the interactions among participants and the security properties depicted in the starting diagram. The extension of choreography diagram includes the security properties not reported by design and indicates how to manage them with additional choreography tasks. The mapping and transformation from choreography diagram to Corda shows how to realize the skeleton of a CorDapp, with the definition of the main objects such as state, contract and workflows.

We implemented a software that automatizes part of the process: from collaboration to choreography diagram with a semi-automatic procedure and then from choreography to the skeleton of a CorDapp. We tested the method through three realistic cases, showing that the method proposed can be used in order to enforce the security properties mentioned above, with particular attention to the first transformation from collaboration to choreography diagram: possible errors are propagated in the next phases and compromises the effectiveness of the contract. Regarding the second phase, the attention is related to the values of attributes in the output states generated by transactions that may not respect the desired evolution of the state. At the same time, they represent only a small sample in order to declare the method working in all contexts and on all types of processes.

Certainly, the method needs to be tested further with a greater pool of samples and with external reviewers, considering supplementing the tests with surveys. Furthermore, some of the activities of mapping and transformation of diagrams, in order to provide an

accepted input to the method, have been performed manually with the presence of experts. For future works, the implementation of the method may include fully automated software that, given a contract described with SecBPMN2BC, produces a complete CorDapp as output ready to be applied in real business contexts. Furthermore, some research works can be carried out analyzing how to overcome the current limitations of the software related, for instance, to the input types including a wide range of acceptable documents.

Bibliography

- [1] O. Altuhhov, R. Matulevičius, and N. Ahmed. An extension of business process model and notation for security risk management. *International Journal of Information System Modeling and Design (IJISMD)*, 4(4):93–113, 2013.
- [2] A. Azaria, A. Ekblaw, T. Vieira, and A. Lippman. Medrec: Using blockchain for medical data access and permission management. In *2016 2nd international conference on open and big data (OBD)*, pages 25–30. IEEE, 2016.
- [3] K. Batoulis, A. Meyer, E. Bazhenova, G. Decker, and M. Weske. Extracting decision logic from process models. In *Advanced Information Systems Engineering: 27th International Conference, CAiSE 2015, Stockholm, Sweden, June 8-12, 2015, Proceedings 27*, pages 349–366. Springer, 2015.
- [4] A. Berry and Z. Milosevic. Extending choreography with business contract constraints. *International Journal of Cooperative Information Systems*, 14(02n03):131–179, 2005.
- [5] K. Bhargavan, A. Delignat-Lavaud, C. Fournet, A. Gollamudi, G. Gonthier, N. Kobeissi, N. Kulatova, A. Rastogi, T. Sibut-Pinote, N. Swamy, et al. Formal verification of smart contracts: Short paper. In *Proceedings of the 2016 ACM workshop on programming languages and analysis for security*, pages 91–96, 2016.
- [6] R. G. Brown, J. Carlyle, I. Grigg, and M. Hearn. Corda: an introduction. *R3 CEV, August*, 1(15):14, 2016.
- [7] V. Buterin et al. A next-generation smart contract and decentralized application platform.
- [8] B. Carminati, E. Ferrari, and C. Rondanini. Blockchain as a platform for secure inter-organizational business processes. In *2018 IEEE 4th International Conference on Collaboration and Internet Computing (CIC)*, pages 122–129. IEEE, 2018.
- [9] Chain. Chain.com, 2014. URL <https://www.chain.com>. Accessed on May 19, 2023.
- [10] O. Choudhury, N. Rudolph, I. Sylla, N. Fairoza, and A. Das. Auto-generation of

- smart contracts from domain-specific ontologies and semantic rules. 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)*, pages 963–970. IEEE, 2018.
- [11] K. Delmolino, M. Arnett, A. Kosba, A. Miller, and E. Shi. Step by step towards creating a safe smart contract: Lessons and insights from a cryptocurrency lab. In *Financial Cryptography and Data Security: FC 2016 International Workshops, BITCOIN, VOTING, and WAHC, Christ Church, Barbados, February 26, 2016, Revised Selected Papers 20*, pages 79–94. Springer, 2016.
- [12] C. Di Ciccio, A. Cecconi, M. Dumas, L. García-Bañuelos, O. López-Pintado, Q. Lu, J. Mendling, A. Ponomarev, A. Binh Tran, and I. Weber. Blockchain support for collaborative business processes. *Informatik Spektrum*, 42:182–190, 2019.
- [13] L. García-Bañuelos, A. Ponomarev, M. Dumas, and I. Weber. Optimized execution of business processes on blockchain. In *Business Process Management: 15th International Conference, BPM 2017, Barcelona, Spain, September 10–15, 2017, Proceedings 15*, pages 130–146. Springer, 2017.
- [14] S. Haarmann, K. Batoulis, A. Nikaj, and M. Weske. Dmn decision execution on the ethereum blockchain. In *Advanced Information Systems Engineering: 30th International Conference, CAiSE 2018, Tallinn, Estonia, June 11-15, 2018, Proceedings 30*, pages 327–341. Springer, 2018.
- [15] S. Haarmann, K. Batoulis, A. Nikaj, and M. Weske. Executing collaborative decisions confidentially on blockchains. In *Business Process Management: Blockchain and Central and Eastern Europe Forum: BPM 2019 Blockchain and CEE Forum, Vienna, Austria, September 1–6, 2019, Proceedings 17*, pages 119–135. Springer, 2019.
- [16] W. Hu, Z. Fan, and Y. Gao. Research on smart contract optimization method on blockchain. *IT Professional*, 21(5):33–38, 2019.
- [17] HYPERLEDGER. An introduction to Hyperledger, 2018. URL http://www.hyperledger.org/wp-content/uploads/2018/07/HL_Whitepaper_IntroductiontoHyperledger.pdf. Accessed on April 2, 2023.
- [18] J. Köpke, G. Meroni, and M. Salnitri. Designing secure business processes for blockchains with secbpmn2bc. *Future Generation Computer Systems*, 141:382–398, 2023.
- [19] W. Labda, N. Mehandjiev, and P. Sampaio. Modeling of privacy-aware business

- processes in bpmn to protect personal data. In *Proceedings of the 29th Annual ACM Symposium on Applied Computing*, pages 1399–1405, 2014.
- [20] C. Lin, D. He, X. Huang, K.-K. R. Choo, and A. V. Vasilakos. Bsein: A blockchain-based secure mutual authentication with fine-grained access control system for industry 4.0. *Journal of network and computer applications*, 116:42–52, 2018.
- [21] Z. Liu and J. Liu. Formal verification of blockchain smart contract based on colored petri net models. In *2019 IEEE 43rd Annual Computer Software and Applications Conference (COMPSAC)*, volume 2, pages 555–560. iee, 2019.
- [22] T. Lodderstedt, D. Basin, and J. Doser. Secureuml: A uml-based modeling language for model-driven security. In *UML 2002—The Unified Modeling Language: Model Engineering, Concepts, and Tools 5th International Conference Dresden, Germany, September 30–October 4, 2002 Proceedings*, pages 426–441. Springer, 2002.
- [23] F. Loukil, K. Boukadi, M. Abed, and C. Ghedira-Guegan. Decentralized collaborative business process execution using blockchain. *World Wide Web*, 24(5):1645–1663, 2021.
- [24] A. Mavridou and A. Laszka. Designing secure ethereum smart contracts: A finite state machine based approach. In *Financial Cryptography and Data Security: 22nd International Conference, FC 2018, Nieuwpoort, Curaçao, February 26–March 2, 2018, Revised Selected Papers 22*, pages 523–540. Springer, 2018.
- [25] J. Mendling, I. Weber, W. V. D. Aalst, J. V. Brocke, C. Cabanillas, F. Daniel, S. Debois, C. D. Ciccio, M. Dumas, S. Dustdar, et al. Blockchains for business process management-challenges and opportunities. *ACM Transactions on Management Information Systems (TMIS)*, 9(1):1–16, 2018.
- [26] S. Nakamoto. Bitcoin: A peer-to-peer electronic cash system. *Decentralized business review*, page 21260, 2008.
- [27] A. Nikaž, K. Batoulis, and M. Weske. Rest-enabled decision making in business process choreographies. In *Service-Oriented Computing: 14th International Conference, ICSOC 2016, Banff, AB, Canada, October 10-13, 2016, Proceedings 14*, pages 547–554. Springer, 2016.
- [28] OMG. Business Process Model and Notation (BPMN) v.2.0, 2011. URL <https://www.omg.org/spec/BPMN/2.0/PDF>. Accessed on March 22, 2023.
- [29] OMG. Decision Model and Notation Version 1.4, 2023. URL <https://www.omg.org/spec/DMN/1.4/PDF>. Accessed on April 21, 2023.

- [30] P. Pullonen, R. Matulevičius, and D. Bogdanov. Pe-bpmn: privacy-enhanced business process model and notation. In *Business Process Management: 15th International Conference, BPM 2017, Barcelona, Spain, September 10–15, 2017, Proceedings 15*, pages 40–56. Springer, 2017.
- [31] R3. Corda fundamental concepts, 2016. URL <https://training.corda.net/corda-fundamentals/concepts>. Accessed on May 17, 2023.
- [32] A. Rodríguez, E. Fernández-Medina, and M. Piattini. Towards a uml 2.0 extension for the modeling of security requirements in business processes. In *Trust and Privacy in Digital Business: Third International Conference, TrustBus 2006, Kraków, Poland, September 4-8, 2006. Proceedings 3*, pages 51–61. Springer, 2006.
- [33] A. Rodríguez, E. Fernández-Medina, and M. Piattini. A bpmn extension for the modeling of security requirements in business processes. *IEICE transactions on information and systems*, 90(4):745–752, 2007.
- [34] N. Szabo. Formalizing and securing relationships on public networks. *First monday*, 1997.
- [35] T. Tateishi, S. Yoshihama, N. Sato, and S. Saito. Automatic smart contract generation using controlled natural language and template. *IBM Journal of Research and Development*, 63(2/3):6–1, 2019.
- [36] P. Tsankov, A. Dan, D. Drachsler-Cohen, A. Gervais, F. Buenzli, and M. Vechev. Securify: Practical security analysis of smart contracts. In *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*, pages 67–82, 2018.
- [37] J. L. Vivas, J. A. Montenegro, and J. López. Towards a business process-driven framework for security engineering with the uml. In *Information Security: 6th International Conference, ISC 2003, Bristol, UK, October 1-3, 2003. Proceedings 6*, pages 381–395. Springer, 2003.
- [38] H. Wang, C. Guo, and S. Cheng. Loc—a new financial loan management system based on smart contracts. *Future Generation Computer Systems*, 100:648–655, 2019.
- [39] I. Weber, X. Xu, R. Riveret, G. Governatori, A. Ponomarev, and J. Mendling. Untrusted business process monitoring and execution using blockchain. In *Business Process Management: 14th International Conference, BPM 2016, Rio de Janeiro, Brazil, September 18-22, 2016. Proceedings 14*, pages 329–347. Springer, 2016.
- [40] R. Yang, R. Wakefield, S. Lyu, S. Jayasuriya, F. Han, X. Yi, X. Yang, G. Amaras-

- inghe, and S. Chen. Public and private blockchain in construction business process and information integration. *Automation in construction*, 118:103276, 2020.
- [41] S. Zareen, A. Akram, and S. Ahmad Khan. Security requirements engineering framework with bpmn 2.0. 2 extension model for development of information systems. *Applied Sciences*, 10(14):4981, 2020.
- [42] A. Zarghami, B. Sapkota, M. Z. Eslami, and M. van Sinderen. Decision as a service: Separating decision-making from application process logic. In *2012 IEEE 16th International Enterprise Distributed Object Computing Conference*, pages 103–112. IEEE, 2012.
- [43] R. Zhang, R. Xue, and L. Liu. Security and privacy on blockchain. *ACM Computing Surveys (CSUR)*, 52(3):1–34, 2019.
- [44] N. Zupan, P. Kasinathan, J. Cuellar, and M. Sauer. Secure smart contract generation based on petri nets. In *Blockchain Technology for Industry 4.0: Secure, Decentralized, Distributed and Trusted Industry Environment*, pages 73–98. Springer, 2020.

List of Figures

2.1	Affiliated medical visit	20
2.2	Secure business processes definition workflow from [18]	21
2.3	Running example with SecBPMN2BC notations	25
2.4	Unilateral and bilateral communication	26
2.5	Choreography diagram of book loan	28
2.6	States shared between nodes of Corda network from [31]	30
2.7	Commands in transactions from [31]	31
3.1	Running example choreography diagram enriched with security properties .	37
3.2	Choreography diagram generated from extended choreography diagram in Figure 3.1	40
4.1	Choreography diagram of affiliated medical visit	47
6.1	Choreography diagram of birth certificate issue	68
6.2	Choreography diagram extended with security notations of birth certificate issue	70
6.3	Choreography diagram from the extended choreography diagram of birth certificate issue	71
6.4	Hospital televisit external choreography diagram	74
6.5	Hospital teleconsultation choreography diagram	75
6.6	Hospital teleconsultation choreography diagram not acceptable	76
6.7	Nodes deployed: citizen, citizen registry birth certificate and notary	82
6.8	Transaction generated by "Request of new birth certificate" workflow . . .	83
6.9	Vaults of participants	84

List of Tables

2.1	Privity spheres	22
2.2	Enforceability of decisions	23
4.1	Mapping Collaboration to Choreography	43
5.1	Conceptual mapping	52

List of Algorithms

5.1	Choreography to Corda	55
5.2	State and contract name definition	56
5.3	Adding nodes to the network	56
5.4	Workflows and state attributes	59
5.5	Commands definition	60
5.6	Commands rules definition	61
5.7	Messages representation in Corda	62
5.8	Exclusive gateways in Corda	63

Listings

6.1	XML document - Birth Certificate Issue	67
6.2	XML document enriched with security properties - Birth Certificate Issue .	69
6.3	XML document of additional tasks - Birth Certificate Issue	70
6.4	XML document of final choreography diagram - Birth Certificate Issue . .	71
6.5	State class - Birth Certificate Issue	80

