Mémoire

Présenté en vue de l'obtention de la
Master de recherche

en Robotique, Informatique et Systèmes de Communication
Data Science and Smart Services

Exécution des itinéraires touristiques avec les Smart Contracts de la blockchain

Presenté par
Rabeb Ben Othmen

Réalisé Au sein de LIFAT Lab

Soutenu en 28 Décembre 2020 devant le jury composé de :

Président :	Sonia Mettali, ,ISAMM - Université de Manouba
Rapporteur :	Hichem Arfa, , ISTIC - Université de Carthage
Co-Encadrant académique :	Amina Brahem, , Université de Tours
Encadrant académique :	Wassim Abbessi, , ISTIC - Université de Carthage

Année Universitaire : 2019-2020

Dedicaces

À mes très chers parents Rached et Wahiba,
À mes soeurs Sara, Safa et Chaima,
À toute ma famille et a tous ces qui m'aiment.

À l'âme de ma grand-mère Aouicha.
Que Dieu lui accorde le paradis.

Remerciements

C'est un plaisir et un moment très agréable de rendre hommage et de formuler des remerciements aux personnes qui, d'une manière ou d'une autre ont apporté leur soutien et contribué à finaliser ce travail.

En particulier, mes remerciements les plus sincères vont à, Mme Amina Brahem, pour sa patience, sa disponibilité, ses précieux conseils qui ont contribué à alimenter ma réflexion et sa supervision éclairée tout au long de la rédaction de ce mémoire de recherche. Je tient a adresser mes profonds remerciements à M. Nizar Messai et M. Yacine Sam pour leur aide et leur suivi pendant toute la durée du stage.

Ma gratitude s'adresse à M. Wassim abbessi pour avoir accepté de me superviser dans cette étude, pour son encouragement, ses conseils judicieux et ses discussions qui m'ont beaucoup aidé au cours de mes recherches.

Mes vifs remerciements aux membres de jury qui ont accepté d'évaluer ce travail.

Je tiens également à exprimer mes remerciements a nos enseignants de l'Institut Supérieur des Technologies de l'Information et des Communications qui ont été toujours à nos côtés et c'est l'occasion de leurs exprimer nos admirations et nos reconnaissances.

Un grand merci à mes parents qui leurs encouragements et leurs soutien m'a permis d'arriver là où je suis maintenant, en croyant toujours en moi et en me soutenant inconditionnellement. Ainsi que mes soeurs pour leurs généreux conseils.

Enfin, un merci sincère à tous mes amis qui ont partagé de nombreux moments avec moi, qui m'ont apporté leur soutien moral et intellectuel tout au long de ce travail.

Table des matières

Dedicaces **i**

Remerciements **iii**

Acronymes **ix**

Introduction Générale **1**

1 Intégration de processus métier sur la blockchain **3**
Introduction . . . 3
1.1 Processus métier . . . 3
1.1.1 Définition de processus métier . . . 4
1.1.2 Gestion des processus métiers . . . 4
1.1.3 Modélisation de processus métier . . . 4
1.2 Blockchain . . . 7
1.2.1 Définition de la blockchain . . . 7
1.2.2 Définition de smart contract . . . 12
1.2.3 Types de la blockchain . . . 13
1.2.3.1 Par rapport à l'accessibilité . . . 13
1.2.3.2 Par rapport à la possession . . . 14
1.3 Exécution des orchestrations des processus métiers sur la blockchain . . . 15
1.4 Exécution des processus métiers collaboratifs sur la blockchain . . . 16
1.4.1 Définition d'un processus métier collaboratif . . . 16
1.4.2 Exécution des processus métiers collaboratifs avec la blockchain . . . 17
1.4.2.1 Collaborations de processus métier . . . 17
1.4.2.2 Chorégraphies de processus métier . . . 18
1.5 Positionnement des approches existantes . . . 19
Conclusion . . . 19

2 Processus métiers transactionnels **20**
2.1 Introduction . . . 20
2.2 Modèles de transaction Avancés (MTA) . . . 20
2.2.1 L'origine de MTA . . . 21
2.2.2 Modèle de transaction flexible . . . 22
2.3 Processus métier transactionnel . . . 24
2.3.1 Flux de contrôle . . . 25
2.3.2 Comportement transactionnel . . . 25
2.3.3 Mécanisme de recouvrement . . . 26
2.4 Exécution des processus métiers transactionnels . . . 27

2.5 Positionnement des approches existantes 27
2.6 Conclusion 28

3 Approche proposée 29
Introduction 29
Partie1 :Smart contract basé sur le processus métier transactionnel 29
3.1 Aperçu de l'architecture de Caterpillar 30
3.2 Solution proposée 33
3.3 Spécification du flux de contrôle 33
3.4 Spécification du flux transactionnel 34
3.4.1 Propriétés transactionnelles d'une activité 34
3.4.2 Cycle de vie des activités 35
3.4.3 Implémentation de mécanismes transactionnels 36
3.4.3.1 Définition de vecteur d'état 36
3.4.3.2 Définitions des fonctions des transitions 37
3.4.3.3 Définition des fonctions d'état de terminaison transactionnelles 37
3.5 Gestion d'échec des exécutions des activités 39
3.5.1 Cas d'échec d'une activité avec la propriété pivot 39
3.5.2 Cas d'échec d'une activité avec la propriété compensable 39
3.5.3 Cas d'échec d'une activité avec la propriété rejouable 40
3.5.4 Cas d'échec d'une activité avec la propriété compensable et rejouable 40
Partie2 : Etendre notre approche dans le cas de TBP collaboratifs s'exécutant sur une blcockain avec permission 40
3.6 Exécution un processus métier transactionnel collaboratif sur une blockchain privée avec permission 41
3.6.1 Aperçu sur SCIP 42
3.6.2 Aperçu sur de BAL 42
3.6.3 Solution proposée 43
3.7 Conclusion 45

4 Mise en oeuvre 46
4.1 Introduction 46
4.2 Platefomes d'éxecution et langages des programmations 46
4.2.1 Cas de blockchain publique sans permission 46
4.2.1.1 Ethereum 47
4.2.1.2 Solidity 48
4.2.2 Cas de blockchain privée avec permission 49
4.2.2.1 Hyperledger Fabric 49
4.2.2.2 JavaScript 50
4.3 Les scénarios implémentés 51
4.3.1 Processus de commande de paquet touristique personnalisé 51
4.3.2 Vente et achat de billet combiné 53
4.4 Conclusion 59

Conclusion Générale 60

Bibliographie 65

Table des figures

1.1 Notation de modélisation de processus métier : catégories d'éléments . . . 5
1.2 Processus métier modélisé comme une orchestration [C.P16] 6
1.3 Processus métier modélisé comme une chorégraphie [C.P16] 6
1.4 Exemple d'une séquence des blocs en blockchain 8
1.5 Structure d'un bloc en blockchain 9
1.6 Transaction avec une signature numérique [WG18] 11
1.7 Les types de blockchain [LIC17] 13

2.1 Un exemple de transaction flexible [Bhi05] 23
2.2 Un exemple de processus métier avec les mécanismes de recouvrement . . . 26

3.1 Architecture de Caterpillar [OLP19] 30
3.2 Compilateur BPMN [OLP19] 31
3.3 Processus d'instanciation de Caterpillar [OLP19] 32
3.4 Cycle de vie d'une activité en fonction des propriétés transactionnelles [Bhi05] 35
3.5 Vecteur d'état 36
3.6 Flux de contrôle d'un exemple d'un modèle BPMN 39
3.7 Modèle de processus métier collaboratif 44
3.8 Communication de TBP avec SC à travers SCIP 44

4.1 Processus de commande d'un packet touristique personnalisé 51
4.2 Exécution de Processus métier transactionnel sur Caterpillar 52
4.3 Concept d'exécution 52
4.4 Vente et achat de billet combiné 53
4.5 Extrait de fichier Docker-compose 54
4.6 Réseau de scénario 55
4.7 Installation de chaincode 56
4.8 Instanciation de chaincode 56
4.9 Conteneur d'instanciation de chaincode 57
4.10 Structure d'application 57
4.11 Wallet Isabella 58
4.12 Vente de ticket combiné 58
4.13 Achat de ticket combiné 59

Liste des tableaux

1.1 Comparaison entre Hyperledger Fabric et Ethereum [PS18] 11

Acronymes

— **BP** : Business process

— **ACID** : Atomicity, Consistance, Isolation, Durability

— **TBP** : Transactional Business Process

— **BPM** : Business Process Management

— **BPMN** : Business Process Model notation

— **BC** : Blockchain

— **SC** : Smart contract

— **ATM** : Advanced Transactional Model

— **FTM** : Flexible Transaction Model

— **BAL** : Blockchain Access Layer

— **SCIP** : Smart Contract Invocation Protocol

— **P2P** : Peer To Peer

— **EVM** : Ethereum Virtual Machine

Introduction Générale

La technologie continue d'émerger et de s'adapter aux entreprises dans divers domaines pour aider et améliorer la performance commerciale de l'entreprise. Par conséquent, des nombreuses entreprises peuvent être confrontées à une décision difficile pour choisir la meilleure solution technique disponible pour atteindre les objectifs d'une entreprise.

Les processus métiers sont un atout essentiel des organisations. Chaque organisation qu'il s'agisse d'une agence gouvernementale ou d'une entreprise doit gérer un certain nombre des processus. Ils intègrent des systèmes, des données et des ressources pour atteindre les objectifs organisationnels en fournissant un service ou un produit à un client. De nos jours, les organisations subissent une pression croissante pour être compétitives et se conformer aux exigences croissantes de la mondialisation, ce qui augmente l'importance des processus métiers que ce soit dans un réseau individuel ou entre multiples organisations qu'on l'appelle les processus inter-organisationnels collaboratifs, un exemple de collaboration de processus métier est l'exécution des itinéraires touristiques. Cependant, le manque de confiance entre les organisations est un obstacle majeur à la mise en oeuvre et à l'exécution de processus métier, ce qui conduit généralement les entreprises à s'appuyer sur des tiers de confiance pour servir des médiateurs.

La technologie Blockchain a un pouvoir potentiel pour exécuter des processus métiers sans compter sur une autorité centrale. Plus précisément, les plates-formes blockchain permettent aux processus métiers collaboratifs ou les processus métiers qui impliquent des participants indépendants d'enregistrer l'état du processus sur un registre infalsifiable et décentralisé, qui stocke et exécute également des smarts contracts pour mettre en oeuvre des transactions dans le registre. La combinaison d'un registre infalsifiable et décentralisé avec les smarts contracts fournit les éléments de base pour mettre en oeuvre ces processus métiers.

En dépit de ce potentiel établi, l'exécution de processus métier en blockchain présente certaines limites. Les entreprises expriment d'un véritable besoin de mécanismes de contrôle pour l'exécution de leurs modèles de processus métier en blockchain afin de concevoir une exécution robuste assurant un traitement fiable et flexible des erreurs et des exceptions avec l'exécution décentralisée et sécurisée. Par ailleurs la sémantique transactionnelle d'un smart contract est très proche du modèle ACID. En effet, il a été admis que les transactions ACID (Atomicité, Consistance, Isolation, Durabilité) se limitent à faire face à des structures de contrôle complexes et à l'exécution à long terme de processus métier (BP).

Pour répondre a notre objectif, notre point de départ était de constater le modèle de processus métier transactionnel (TBP) pour assurer un certain niveau de correction des exécutions. En effet les TBP sont apparus comme une extension des transactions ACID pour surmonter ces limites. Plus précisément, TBP présente l'ensemble des processus métiers avec les modèles transactionnels avancés (MTA). L'un de modèle de MTA est le modèle de transaction flexible (FTM) qui nous avons l'utilisé dans notre travail. Ce modèle englobe le mécanisme transactionnel tel que les alternatives et la compensation.

L'objectif de notre travail est de proposer une approche pour assurer une exécution fiable des processus métiers en blockchain. Pour ce faire, nous avons choisi d'enrichir la description de processus métier avec des propriétés transactionnelles pour mieux exprimer leurs comportements. Ensuite, nous avons exploité un modèle de processus métier dans un système blockchain. On peut considérer notre modèle comme une transaction structurée où les activités sont des sous-transactions et les interactions sont des dépendances transactionnelles. Notre approche est développée en deux parties, dans la première partie nous développons notre processus métier qui modélise l'exécution d'un itinéraire touristique en blockchain publique sans permission, a ce stade nous améliorons le comportement transactionnel du processus métier en assurant certaines règles de cohérence. Dans la deuxième partie nous étendons notre approche dans le cas d'une collaboration de processus métier modélisant l'achat et la vente d'un billet électronique exécuté dans une blockchain privée avec permission. L'originalité de notre approche à ce niveau est la flexibilité que nous offrons aux entreprises pour spécifier leurs besoins en terme de structure de contrôle, de correction, de décentralisation et de confiance.

Notre document est organisé comme suit : Dans le chapitre 1 nous présentons les deux technologies principales où se concentre notre travail qui sont les processus métier et la blockchain avec un aperçu sur les travaux existant dans le domaine d'intégration de processus métier en blockahain, nous précisons ce qui manque dans ces approches.
Dans le chapitre 2, nous présentons les processus métiers transactionnels, nous montrons le modèle de transaction avancé ainsi que le modèle transactionnel avancé.
Nous présentons ensuite la positionnement des approches antérieurs par rapport a notre solution.
Le chapitre 3 constitue le coeur de notre travail où nous présentons notre approche en deux parties.
Dans le chapitre 4, nous présentons les plateformes utilisés dans notre approche, puis nous illustrons l'implantation de notre scénarios implémentés dans les environnements de blockchain Ethereum et Hyperledger Fabric.
Finalement, le chapitre 5 dresse la conclusion de notre travail et les éventuelles perspectives.

Chapitre 1

Intégration de processus métier sur la blockchain

Introduction

Plusieurs thématiques de recherche ont abordé les problèmes de la confiance dans le domaine du processus métier.
Un grand nombre des transactions se produisent entre les entités commerciales. Ces transactions peuvent être prises en charge via le réseau blockchain, ce qui peut favoriser l'échange des valeurs de manière efficace et économique. Par conséquent les plateformes de gestion des processus métiers peuvent être soutenues par une intégration avec des réseaux blockchain pour garantir la sécurité, la confiance et la transparence de toutes transactions effectuées.
Dans ce chapitre nous présentons la notion de processus métier, un aperçu sur la technologie de la blockchain puis nous citons quelque approches qui discutent l'exécution des processus métiers sur la blockchain en étudiant leur capacité à intégrer les dimensions de notre problématique.

1.1 Processus métier

La notion de processus métier joue un rôle clé dans le développement des systèmes d'information. Ils sont généralement considérés comme un préalable nécessaire pour concevoir la dynamique du système d'information d'une organisation en fournissant des processus métiers robustes qui s'adaptent à leurs activités.
De plus, ces processus métier représentent l'un des principaux atouts des organisations pour des nombreuses raisons. Ils ont un impact direct sur l'attractivité des produits et services, influencent l'expérience client et finalement le chiffre d'affaires dans le cas des entreprises. En outre, ils orchestrent les ressources de l'entreprise pour répondre à ces demandes externes, ils sont donc un facteur clé déterminant le coût de service et l'efficacité opérationnelle. Par conséquent, tout échec de processus peut paralyser la vie de l'entreprise et l'ensemble de l'écosystème de processus [MD18].
La suite de cette section présente la définition de processus métier ainsi que ses méthodes de modélisation et son outil de déploiement.

1.1.1 Définition de processus métier

Un processus métier ou "Business Process" en anglais est défini par le Workflow Management Coalition (WfMC) comme un ensemble des activités ou des procédures liées les unes aux autres collectivement pour atteindre un objectif métier en définissant les interactions fonctionnelles dans la structure organisationnelle [KHE13].

En effet un processus métier est une chorégraphie d'activités et des tâches organisées qui permettent d'atteindre un objectif organisationnel et un résultat précis. Autrement dit, il est considéré comme un ensemble des relations logiques entre un groupe des activités, y compris une interaction entre participants sous la forme d'échange d'informations tel que des acteurs humains, des organisations, des applications, des documents ou d'autres sources d'informations.

1.1.2 Gestion des processus métiers

La gestion des processus métiers BPM "Business Process Managenement" est définie par [VDA03] comme l'utilisation de processus métier à l'aide des méthodes, des techniques et des outils pour concevoir contrôler, exécuter et analyser des processus opérationnels faisant intervenir des humains, des organisations, des applications, des documents et d'autres sources d'information. Cela est donc considéré comme une approche qui permet aux organisations de faire assurer la mise en oeuvre efficace de ses processus tout en répondant aux besoins de ses différents interlocuteurs avec le meilleur niveau des performances et de maîtrise des coûts.
Dans une autre définition La BPM est considérée comme une extension des systèmes et des approches classiques du système de gestion des flux de travail (WFMS) où il est défini comme un système qui définit, créé et gérer l'exécution des flux de travail grâce à l'utilisation des logiciels, s'exécutant sur un ou plusieurs moteurs de flux de travail, qui est capable d'interpréter la définition de processus, d'interagir avec les participants de flux de travail et d'invoquer l'utilisation des outils et des applications informatiques.

1.1.3 Modélisation de processus métier

La modélisation des processus métiers permet d'exprimer la fonction du processus en définissant toutes les activités à réaliser et leur séquence d'exécution. Pour cette raison, il existe deux méthodes dans la littérature pour définir le comportement du processus une approche impérative et une approche déclarative [LA11].
L'approche impérative se concentre sur la définition précise de la manière dont l'ensemble d'activités sera réalisé. Pour cela, un ensemble de liens ou des connecteurs est utilisés pour décrire clairement la séquence d'exécution entre les différentes activités. Tandis que l'approche déclarative se concentre sur «ce qu'il faut faire» au lieu de «quoi faire». Cela maintient le plan d'exécution implicite dans la phase de modélisation du processus, et en même temps évite de lister explicitement tous les plans d'exécution possibles dans cette phase.

Le standard BPMN 2.0

Le langage BPMN "Business Process Modeling Notation" est un standard de modélisation des processus métiers qui permet de définir une notation graphique commune pour

tous les outils de modélisation. Il est proposé par l'OMG/BMPI "Object Management Group et le Business Process Management Initiative" depuis leur fusion en 2005. L'objectif principal du BPMN est de fournir une notation facile à comprendre par tous les acteurs d'une entreprise depuis les analystes métier qui crée les ébauches initiales des procédures, aux développeurs techniques responsables de la mise en oeuvre de la technologie qui va exécuter ces processus [Wes19].

D'un point de vue théorique des graphes, la figure 1.1 présente les éléments de notation dans les diagrammes de processus métier sont divisés en quatre catégories de base, dont chacune se compose d'un ensemble d'éléments [Wes19] :

Les flux des objets sont les éléments constitutifs des processus métiers ; ils comprennent des événements, des activités et des passerelles.
Les swimlanes sont limités à une hiérarchie à deux niveaux : les pools et les lanes. Les pools représentent des organisations qui participent à l'interaction de plusieurs processus métier, chacun étant mis en oeuvre par une organisation. et les lanes représentent les entités organisationnelles telles que les services au sein d'une organisation participante.
Les artefacts comme les objets des données et les annotations de texte ou groupes sont utilisés pour afficher des informations supplémentaires sur un processus métier qui ne sont pas directement pertinentes pour les flux de séquence ou les flux de message du processus.
Les objets de connexion comprennent le flux de séquence, les flux de message et les associations sont utilisé pour connecter des objets de flux, des lanes ou des artefacts.

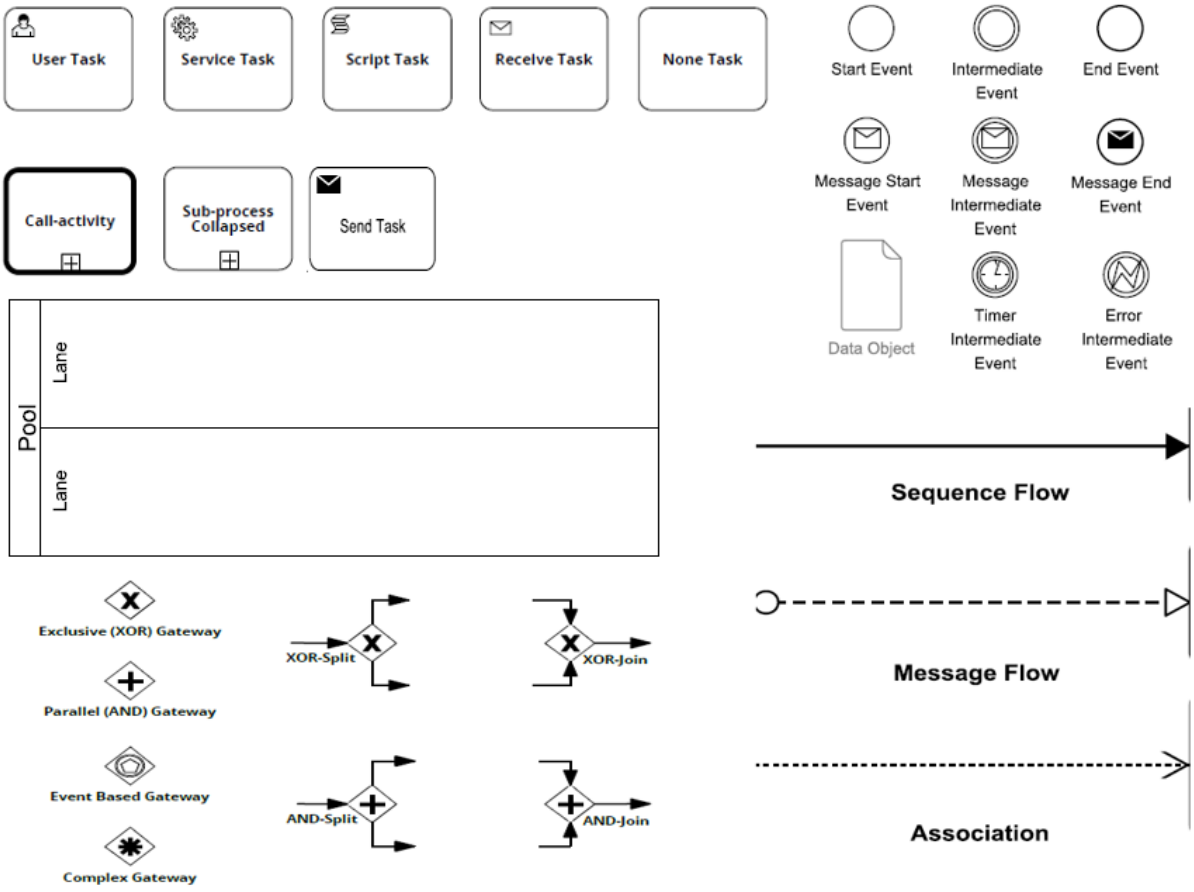


FIGURE 1.1 – Notation de modélisation de processus métier : catégories d'éléments

Un processus BPMN peut également être décrit avec trois catégories de processus [AS08] : Orchestration, Chorégraphie et collaboration.

Orchestration Les modèles d'orchestration bpmn nécessitent à impliquer une perspective de coordination unique, ils représentent la vision du processus par une entreprise ou une organisation spécifique en tant que tel, un processus d'orchestration décrit comment une seule entité commerciale gère les choses. Cependant un modèle bpmn peut contenir plus qu'une orchestration. Si tel est le cas, chaque organisation apparaît dans son propre conteneur (par exemple, le pool dans BPMN 2.0) peuvent être définis comme des exemples d'orchestration [AS08].
La figure 1.2 présente un modèle de processus métier modélisé comme une orchestration.

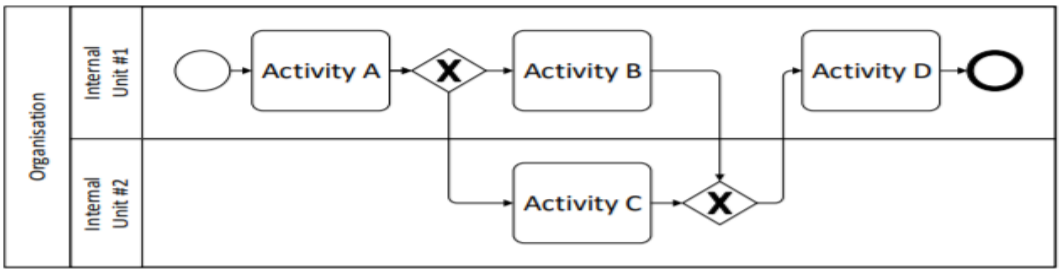


FIGURE 1.2 – Processus métier modélisé comme une orchestration [C.P16]

Chorégraphie Un modèle de processus de chorégraphie est une définition du comportement attendu (un type de contrat ou protocole procédural) entre les participants en interaction. Dans un modèle BPMN une chorégraphie définit la séquence des interactions entre deux ou plusieurs participants. Ces interactions sont les communications sous forme des échanges des messages entre les participants [AS08].
La figure 1.3 présente un modèle de processus métier modélisé comme une chorégraphie.

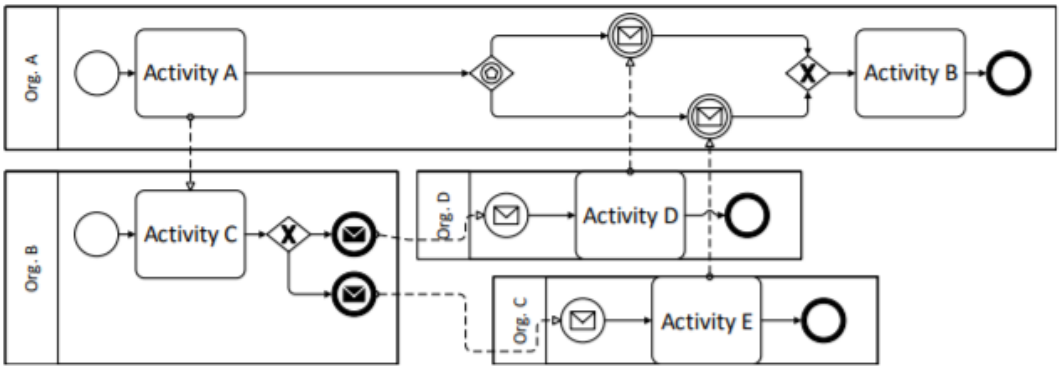


FIGURE 1.3 – Processus métier modélisé comme une chorégraphie [C.P16]

Collaboration La collaboration a une signification particulière dans BPMN où une chorégraphie définit l'ensemble ordonné (un protocole) d'interactions entre les participants, une collaboration montre simplement les participants et leurs interactions. Elle peut également contenir une chorégraphie (lorsqu'elle est disponible dans bpmn) et une ou plusieurs orchestrations pour être plus spécifique elle est n'importe quel diagramme BPMN qui contient deux participants ou plus.

1.2 Blockchain

La blockchain est une technologie révolutionnaire, elle a récemment expérimenté une large éventail d'application blockchain allant de la crypto-monnaie, des services financiers, de la gestion des risques, de l'Internet des objets aux services publics et sociaux.
Cette section fournit un aperçu sur la Blockchain, qui comprend la définition de la blockchain y compris la vérification des transactions, les consensus et l'architecture des blocs, la définition des smart contracts et les types de la blockchain.

1.2.1 Définition de la blockchain

La blockchain [ZZ17, WH19] est une infrastructure mondiale ouverte qui permet aux particuliers d'effectuer des transactions sans organe central de contrôle de manière sécurisé et avec un coût réduit. Elle implique des technologies tels que peer-to-peer (P2P), la tolérence byzatine, les smart contracts et les algorithmes des consensus.

L'article [Dre17] définit la blockchain comme un système des registres peer-to-peer décentralisé avec domaine d'application commun. Le contenu de l'information est ordonné, et connecté entre chaque bloc des données structurées qui suivent certaines réglementations. La technologie de sécurité est combinée entre la preuve de cryptographie et un algorithme de hachage qui est utilisé pour obtenir et préserver l'intégrité, en ne permettant pas d'effectuer des modifications internes ou externes. La définition de la blockchain est ambiguë, elle peut s'appliquer à différents cas tels que technologiques, commerciaux ou juridiques.
Les blockchains peuvent être considérées comme une zone basée sur la confiance, un support d'échange, un canal sécurisé, un ensemble des capacités décentralisées, et bien plus encore [Mou16].

D'un point de vue économique, le réseau blockchain est efficace puisqu'il réduit le doublement des opérations et élimine le besoin d'intermédiaire. Elle est moins vulnérable car les modèles de vérification des informations entrantes sont fiables. Les transactions dans la blockchain sont vérifiables et authentifiées avec les mêmes participants dans les systèmes de transaction lorsque l'enregistrement des systèmes est partagé et disponible pour toutes les parties impliquées [Gup17].

La technologie blockchain est composée de six éléments clés [ZZ17, LIC17] :

- **Décentralisation :** La blockchain est un réseau peer-to-peer décentralisé signifie que la blockchain n'a plus besoin de s'appuyer sur un noeud centralisé, les données peuvent être enregistrées, stockées et mises à jour de manière distribuée.
- **Transparent :** L'enregistrement des données par le système blockchain est transparent pour chaque noeud, il est également transparent lors de la mise à jour des données, c'est pourquoi la blockchain peut être fiable.
- **Open source :** La plupart des systèmes de blockchain sont ouverts à tout le monde, l'enregistrement peut être vérifié et les gens peuvent également utiliser la technologie de blockchain pour créer n'importe quelle application de leur choix.
- **Persistance :** Les transactions peuvent être validées rapidement. Une fois qu'une transaction est enregistrée dans la blockchain, il est presque impossible de supprimer

ou d'annuler la transaction. Les blocs contenant des transactions invalides peuvent être trouvés immédiatement.

— **Autonomie :** En raison de la base du consensus, chaque noeud du système blockchain peut transférer ou mettre à jour des données en toute sécurité, l'idée est de confier une seule personne à l'ensemble du système.

— **Anonymat :** Les technologies blockchain ont résolu le problème de confiance entre noeuds à noeuds, de sorte que le transfert des données ou même la transaction peut être anonyme, il suffit de connaître l'adresse blockchain de la personne.

La représentation de haut niveau du cadre de la blockchain, est peut être divisée en couche réseau qui concerne le réseau peer-to-peer et la consensus de la blockchain, la couche de données est l'unité fondamentale de la blockchain qui définie la structure de données et la couche d'application présente les différentes applications qui peuvent intégrer la blockchain pour tirer parti de son grand livre continu y compris les smart contract et la cryptographie.

Les éléments de blockchain

La blockchain est une structure des données, un moyen de stocker des informations. De plus, elle est un protocole de transfert de valeur. Elle compose des parties qui jouent un rôle dans l'écosystème de la blockchain.

Les pairs (peers)

Les ordinateurs qui participent au réseau et sont disponibles pour tous les autres membres du réseau, tous basés sur l'égalité des droits des participants sans exception, et tous les noeuds qui fournissent le réseau du client et du serveur, décrivent le terme d'égal à égal. Les réseaux de système peer-to-peer sont décentralisés [Ant17] et les applications de système P2P comprennent des fonctionnalités telles que la garantie de la confidentialité, le partage des données et la distribution de contenu. Ces applications transforment les ordinateurs des utilisateurs au sein du réseau en noeuds dont tout le système distribué se compose [Dre17].

Les Blocs

De point de vue de la structure des données, la blockchain est une liste d'enregistrements des blocs qui sont liés et sécurisés par l'application de cryptographie comme présente la figure 1.4.

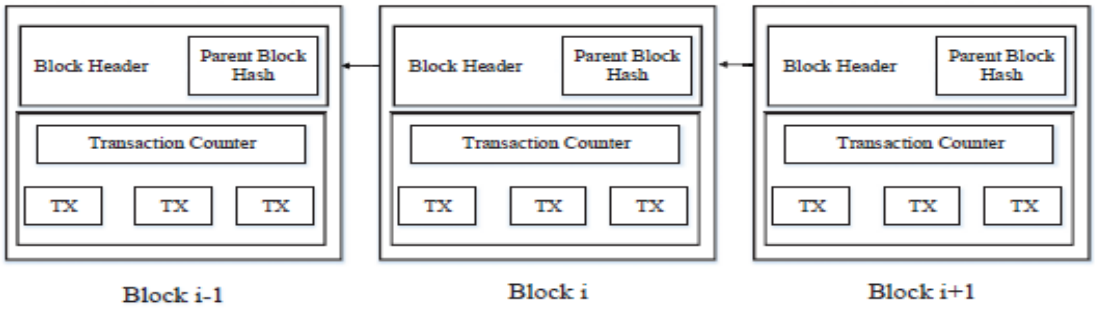


FIGURE 1.4 – Exemple d'une séquence des blocs en blockchain

Les blocs sont enchaînés de manière à ajouter l'en-tête de chaque bloc inclut un hachage cryptographique des transactions du bloc, ainsi qu'une copie du hachage de l'en-tête du bloc précédent. Le premier bloc d'une blockchain est appelé bloc genesis puisque il n'a pas de bloc parent [ZZ17].
chaque bloc est une structure des données des conteneurs de taille fixe, il se compose de l'en-tête de bloc et de corps du bloc [ZZ17] comme illustré dans la figure 1.5 .

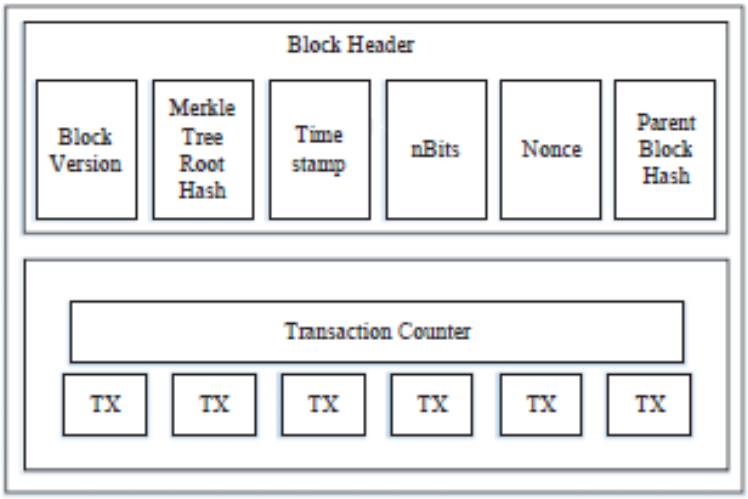


FIGURE 1.5 – Structure d'un bloc en blockchain

L'en-tête de bloc comprend généralement : version de bloc pour les règles de validation de bloc suivre, le hachage de bloc précédent, le hachage de la racine de l'arbre Merkle regroupant tous les hachages des transactions incluses, l'horodatage(timestamp) pour la traçabilité, nBits (la cible de hachage actuelle) et la Nonce utilisé pour le consensus.

Corps de bloc comprend généralement le compteur des transactions et toutes les transactions, le nombre maximum des transactions qu'un bloc peut contenir dépend de la taille de bloc et de la taille de chaque transaction.

Les noeuds

Les noeuds sont l'unité principale du réseau, chaque noeud peut être considéré comme un ordinateur individuel. Ils ont les mêmes capacités fonctionnelles et responsabilités [Dre17]. Chaque noeud du système collecte, stocke et transfère des informations à tous les autres noeuds. Chaque noeud du réseau peer-to-peer collecte des nouvelles données de transaction dans une boîte de réception et sélectionne des nouveaux blocs pour un traitement ultérieur. Les nouveaux blocs traités instantanément a l'aide d'un mécanisme de consensus. Chaque noeud du réseau gère les enregistrements des transactions récentes en les approuvant pour l'autorisation, l'exactitude de la configuration et la précision sémantique et collecte uniquement des données des transactions valides pour crée un nouveau bloc. Une fois le nouveau bloc créé, il est envoyé à tous les autres noeuds du système pour la prochaine vérification [Dre17].

Algoritmes de consensus de blockchain

Il peut arriver que dans un certain moment, plusieurs noeuds parviennent à générer un nouveau bloc simultanément, tant qu'il n'y a pas d'autorité centrale pour garantir que les registres sur les noeuds distribués sont tous les mêmes, certains protocoles sont nécessaires pour garantir la cohérence des registres dans les différents noeuds [ZZ17, DPD18].
Avec la stratégie de consensus, la technologie Blockchain peut résoudre efficacement le problème de concurrence de manière complètement distribuée. Au lieu d'avoir une autorité centrale qui gère une base des données et en garde l'authenticité, une copie de l'ensemble de la base des données est distribuée à chaque noeud de réseau.
Ces noeuds suivent le protocole de consensus et comparent leurs versions ensemble à travers un processus continu de vote se mettre d'accord sur la validité des transactions. La version qui obtient le plus de votes du réseau est acceptée comme authentique et le processus se répète indéfiniment [SA17].
Divers mécanismes de consensus ont été conçus, notamment la preuve de travail (PoW), la preuve d'enjeu (PoS), la tolérance aux pannes byzantine pratique (PBFT).

Les transactions

Les transactions sont un groupe d'opérations de structure des données séquentielles combinées logiquement, qui chiffrent le transfert de valeur entre les participants de système [Ant17].
La transaction a une vérification de la propriété de chaque quantité de données d'entrée présentées dans le schéma d'une signature numérique du propriétaire, qui peut être confirmé par n'importe qui indépendamment [Ant17].

Validation des transactions La blockchain utilise un mécanisme de cryptographie asymétrique pour valider l'authentification des transactions [ZZ17, WG18].

La cryptographie est basée sur des fonctions mathématiques qui permettent de créer des codes numériques et des signatures numériques durables dans le système [Ant17]. Le besoin de cryptographie est déterminé en assurant la sécurité des données, y compris l'identification, l'autorisation et l'authentification [Dre17].

La signature numérique est un accord avec le contenu des données de transaction [Dre17]. Comme l'équivalent des signatures manuscrites, des signatures numériques ont été créées, elles utilisent le hachage basé sur la cryptographie, ainsi qu'un flux d'informations privé-public de cryptographie asymétrique.
La possibilité d'ajouter des transactions à la blockchain est activée par la signature numérique. l'utilisation des signatures numériques garantit la provenance des transactions d'un utilisateur à un autre, chaque transaction étant connectée aux transactions précédentes.

La figure suivante 1.6 illustre les transactions avec des signatures numériques dans la blockchain. Chaque utilisateur possède une paire de clé privée et de clé publique. La clé privée qui doit être gardée confidentielle. Elle est utilisée pour signer les transactions. Puis les transactions signées numériques sont diffusées sur l'ensemble de réseau.

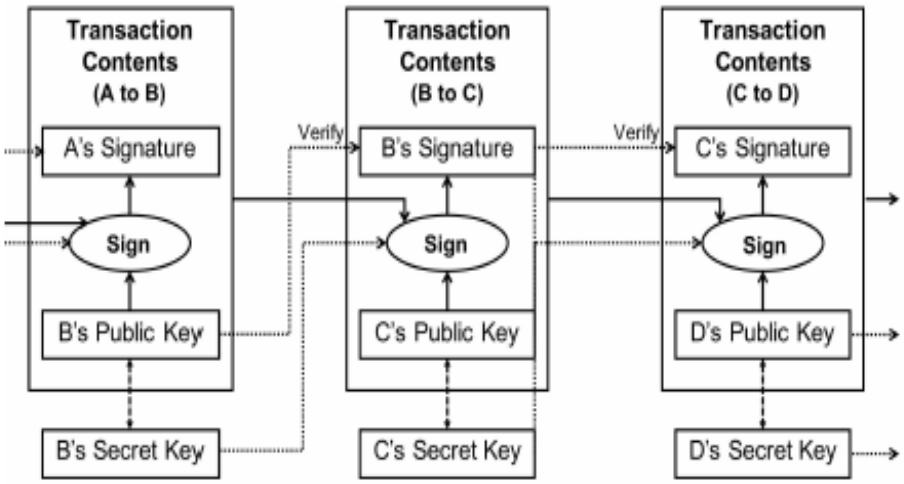


FIGURE 1.6 – Transaction avec une signature numérique [WG18]

La signature numérique typique comporte deux phases une phase de signature et une phase de vérification. Dans la phase de signature la clé privée est utilisée pour crypter les données avant de les envoyer à la contrepartie et dans la phase de vérification la clé publique est utilisée pour valider la transaction reçue.

Les plateformes de blockchain

Afin de mettre en oeuvre une nouvelle technologie dans une entreprise, il est essentiel de choisir la bonne méthode pour y parvenir. Il existe des nombreuses plates-formes de développement de blockchain existantes et prometteuses parmis elles Ethereum et Hyperledger qui sont les plus populaires et les plus couramment utilisés.
Ethereum est est une plates-formes de blockchain à usage général, qui permet aux utilisateurs d'écrire leur propre code algorithmique et d'exécuter des processus logiques personnalisés, elle n'est pas liée à une application spécifique.

Hyperledger est une plateforme blockchain spécifique qui est optimisée pour une tâche spécifique comme le suivi des actifs et le transfert des valeurs [PS18].
Une comparaison plus détaillée peut être vue dans le tableau 1.1.

TABLE 1.1 – Comparaison entre Hyperledger Fabric et Ethereum [PS18]

Charactéristique	Ethereum	Hyperledger Fabric
Description	Platforme génerique	Plateforme modulaire
Gouvernance	Développeur Ethereum	Fondation Linux
Mode d'opération	Sans permission, publique ou privée)	Avec permission privée
Consensus	Proof-Of-Work	practical Byzantine Fault Tolerance
Etat	Base de données clé-valeur	Données de compte
Devise	Ether	Aucun
Transaction	Anonyme ou privé	publique ou confidentiel
Smart contract	smart contract (solidity)	Chaincode (Java,Go)

1.2.2 Définition de smart contract

Le concept de smart contract a été introduit par Nick Szabo en 1994 [Sza94] qui l'a défini comme un protocole de transaction informatisé qui exécute les termes d'un contrat. Les objectifs généraux de la conception des smart contracts sont de satisfaire les conditions contractuelles courantes (telles que les conditions de paiement, les privilèges, la confidentialité et même l'exécution), de minimiser les exceptions à la fois malveillantes et accidentelles et de minimiser le besoin d'intermédiaire de confiance. Les objectifs économiques connexes comprennent le coût de mise en application et d'autres coûts de transaction.

Selon [AB19] les smart contracts sont rédigés sous forme des programmes informatique définissent des accords informatisés qui permettent d'exécuter automatiquement des conditions définies au préalable et inscrites dans la blockchain.
Ils permettent d'effectuer des transactions entre des parties anonymes ou non approuvées sans avoir besoin d'une autorité centrale concernant une vente de soins valide ou une évaluation de prêt ou un vote ou un suivi des soins de santé, etc. Chaque transaction publiée dans un réseau blockhain est correspondante à un smart contract (la création d'un smart contract est un type de transaction). Une fois ce smart contract est créé, il est affecté à une adresse unique. cette adresse présente un identifiant sécurisé, résultat d'opérations mathématiques et de l'application d'algorithmes de chiffrement.
Un contrat peut être considéré comme une classe des concepts orientés objet et chaque déploiement du contrat peut être considéré comme une instance de l'objet. Il peut également être déployé sur un réseau plusieurs fois, et chaque instance avoir une adresse distincte.

Pour un large éventail des applications potentielles, les smart contract ont pour objectif de fournir plus de sécurité, de réduire les coûts et d'augmenter l'efficacité des processus en réduisant les délais par rapport aux contrats de droit traditionnel. Dans une solution Blockchain, un utilisateur interagit avec une application blockchain ou une autre interface utilisateur et appelle des appels dans le smart contract via un kit de développement logiciel (SDK).

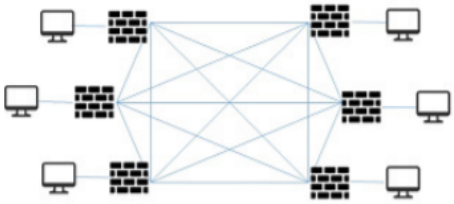
Les smart contracts peuvent être développés et déployés dans différentes plateformes de blockchain. Ces différentes plateformes ont des caractéristiques différentes citons ces deux plateformes typique ethreum et hyperledger.

Ethereum Dans Ethereum le smart contract est écrit dans un langage de programmation spécifique «Solidity» Il est ensuite compilé en «bytecode» qui est lu et exécuté sur la machine virtuelle de ethereum (EVM).
Un contrat peut définir plusieurs points d'entrée d'exécution. Dans le langage Solidity d'Ethereum, chaque point d'entrée est défini comme une fonction.

Hyperledger Hyperledger Fabric exploite la technique des conteneurs pour héberger des smart contracts appelés «chaincode» et écrite en Go, Javascript ou java qui comprennent la logique applicative de système. En plus de cela, «chaincode» est la seule canal qui interagit avec la blockchain et la seule source qui génère les transactions[SW18].

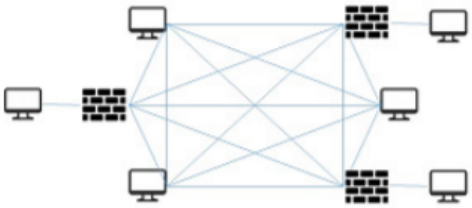
1.2.3 Types de la blockchain

Les systèmes de blockchain actuels peuvent être classés en deux catégories : blockchain basées sur l'accès au traitement des transactions (avec permission, sans permission ou consortium) et blockchain basé sur l'accès aux données (public ou privé) [DPD18, Gro16].



(a) Blockchain privée/avec permission

(b) Blockchain publique/ sans permission



(c) Blockchain consortium

FIGURE 1.7 – Les types de blockchain [LIC17]

1.2.3.1 Par rapport à l'accessibilité

En terme d'accès aux données la blockchain peut être diviser en deux type : blockchain privée et blockchain publique [DPD18, Gro16, LIC17].

Blockchain privée

Une blockchain privée est une blockchain dans laquelle l'accès direct aux données de la blockchain et la validation des transactions sont limités à une liste prédéfinie des entités. Elle a une gestion stricte en ce qui concerne l'autorité d'accès aux données dans le réseau. Aucun des noeuds de réseau ne peut participer à la vérification et la validation des transactions. Ce type de blockchain est établi pour faciliter le partage privé et l'échange des données entre un groupe des individus (au sein d'une organisation) ou plusieurs organisations appartenant à la même branche.
La Figure 1.7(a) montre une blockchain privée qui est égalementune blockchain avec permission.

Blockchain publique

Une blockchain publique est une blockchain dans laquelle il n'y a aucune restriction sur la lecture des données blockchain (qui peuvent être cryptées) et la validation des transactions pour inclusion dans la blockchain. Tout le monde peut vérifier la transaction, et peut également participer au processus d'obtention d'un consensus. L'avantage du réseau publique est l'anonymat de l'utilisateur et la transparence totale du registre.
La Figure 1.7(b) montre La blockchain publique qui est également une blockchain sans permission.

1.2.3.2 Par rapport à la possession

En terme de contrôle d'accès, la blockchain peut être divisée en deux catégories : «avec permission » et «sans permission». Leurs différences ne sont pas seulement liées à la capacité des noeuds à lire, écrire et valider des transactions, mais également à la mesure dans laquelle les noeuds peuvent participer au processus de consensus [KW18, DPD18].

Blockchain avec permission

Une blockchain avec permission est une blockchain, dans laquelle le traitement des transactions est effectué par une liste prédéfinie de sujets avec des identités connues.
Tout les utilisateurs inconnus ne pourront pas y accéder que par invitation. La participation des noeuds est déterminée soit par un ensemble des règles ou par un réseau qui contrôle l'accès. Une entité centrale décide et attribue le droit à des pairs individuels de participer aux opérations d'écriture ou de lecture de la blockchain.
Pour assurer l'encapsulation et la confidentialité, le lecteur et l'écrivain peuvent également fonctionner dans des chaînes des blocs parallèles séparées qui sont interconnectées. L'instance le plus connue de la blockchain avec permission est Hyperledger Fabric (de Linux Foundation).

Blockchain sans permission

Une blockchain sans permission est une blockchain dans laquelle il n'y a pas des restrictions sur les identités des processeurs de transaction (c'est-à-dire les utilisateurs éligibles pour créer des blocs des transactions). Chaque participant dispose d'une autorisation complète pour lire et écrire des transactions, effectuer des audits dans la blockchain ou en visualiser une partie et tout pair peut rejoindre et quitter le réseau en tant que lecteur et écrivain à tout moment.
Cependant, avec l'utilisation de primitives cryptographiques, il est techniquement possible de concevoir une blockchain sans permission qui cache les informations pertinentes pour la confidentialité. Ethereum et Bitcoin sont des exemples qui utilisent une approche où tout le monde peut lire et écrire.

Blockchain consortium

C'est un système hybride entre Blockchain publique et privée. La blockchain consortium présente dans la figure 1.7(c) peut être considérée comme une blockchain partiellement privée et avec permission, où il n'y a pas d'organisation responsable de consensus et de la vérification du bloc, mais un ensemble des noeuds prédéterminés.
Pour la vérification de bloc, un schéma multi-signature est utilisé, dans lequel le bloc est considéré comme valide uniquement lorsqu'il est signé par ces noeuds. Ainsi, il s'agit d'un système partiellement centralisé, de fait de contrôle par certains noeuds de validation sélectionnés, contrairement à la blockchain privée, qui est totalement centralisée, et à la blockchain publique, qui est complètement décentralisé.

1.3 Exécution des orchestrations des processus métiers sur la blockchain

Une des processus métiers d'orchestration est la gestion de la chaîne d'approvisionnement où nous pouvons trouvé certaines installations, telles que fournisseur, usine, centre de distribution, etc., à travers lesquels, le matériel, l'argent et l'information circulent. Certaines transactions ont lieu à ces interfaces. Par exemple, entre le fournisseur et l'usine, entre l'usine et le centre de distribution. Les problèmes liés à la confiance sont inévitables en raison de l'existence de licenciements liés aux finances dans la tenue de registres, la technologie blockchain est attendue pour accélérer les processus et les rendre plus fiables. Un bon nombre des études récentes ont étudié l'impact de la blockchain sur ces types de processus. L'accent a été mis sur des éléments tels que le coût, la qualité, la réduction des risques et souplesse [SK19].

Des travaux récents ont utilisé la technologie blockchain comme infrastructure d'exécution [SA16] pour imposer un certain ordre d'exécution prédéfini des tâches BP. Abeyratne et al. ont discuté du bénéfice potentiel de technologie blockchain dans la chaîne d'approvisionnement de fabrication. Ils ont proposé que les caractéristiques héritées de la blockchain renforcer la confiance grâce à la transparence et à la traçabilité dans toute transaction de données, de biens et de ressources financières. Et il pourrait offrir une plate-forme innovante pour les nouveaux mécanisme de transaction transparent dans les industries et les entreprises. En outre, ils utilisent également un exemple pour démontrer comment la technologie blockchain peut être utilisée dans un réseau de chaîne d'approvisionnement mondial.

Dans le domaine de finance Du et all. [MD20] ont construit une plateforme financière basée sur la blockchain pour gérer le modèle de financement. Leur plateforme vise à faciliter confiance entre les participants, ainsi que garantir l'efficacité financière, la circulation des connaissances et la disponibilité des services financiers. Les auteurs ont également proposé d'utiliser le cryptage homomorphique dans la blockchain pour assurer la confidentialité des utilisateurs.

En outre, plusieurs études se sont concentrées sur l'analyse des dimensions comportementales importantes qui affectent l'adoption de la blockchain dans le supply chain comme [KF18]. L'objectif de cet article est d'explorer l'adoption des utilisateurs des technologies blockchain dans les applications de traçabilité de la chaîne d'approvisionnement en utilisant la théorie unifiée de l'acceptation et de l'utilisation de la technologie (UTAUT) comme cadre pour l'acceptation de la technologie et la construction de la confiance dans la technologie de l'information. Cette théorie fournit un cadre conceptuel robuste pour expliquer ces relations et soutiennent le développement d'outils blockchain. Un modèle conceptuel est présenté théoriquement supporte des propositions de recherche équilibré avec les implications de la gestion de la chaîne d'approvisionnement comme cadre potentiel pour comprendre l'adoption de la blockchain pour la traçabilité de processus.

1.4 Exécution des processus métiers collaboratifs sur la blockchain

On prétend que la blockchain a montré son grand potentiel pour innover les processus métier dans les industries et les domaines publics en remplaçant les autorités centralisées par des systèmes distribués.

En exécutant un processus inter-organisationnel sous forme de smart contract sur la blockchain, certains problèmes courants peuvent être résolus. Premièrement, la blockchain servira de registre immuable et fiable pour enregistrer et valider chaque opération pendant l'exécution du processus. Les transactions peuvent être limitées à un ensemble de rôle, c'est-à-dire que les transactions ne peuvent être exécutées que par des participants avec des signatures autorisées [OLP18].

En outre, d'autres règles métier peuvent être mises en oeuvre par les smart contracts pour effectuer les tâches. Les participants peuvent accéder aux transactions générées par le processus qui permettent le suivi de son exécution.
Cette section donne une définition sur les processus collaboratifs de manière générale avec une étude sur les recherches effectuées qui reposent sur la traduction de modèles de processus capturés dans le Business Process Model et la notation (BPMN) en smart contract.

1.4.1 Définition d'un processus métier collaboratif

Un processus métier collaboratif est un processus qui implique des entreprises partenaires, et se compose des parties correspondant à chaque partenaire. Son objectif principal est de définir la chorégraphie des interactions entre les organisations impliquées dans la collaboration [LA18].
Un processus métier collaboratif peut être caractérisé par :

- Plusieurs identités indépendantes qui représentent plusieurs décideurs.
- Plusieurs moteurs de workflow, cela signifie une architecture décentralisée.
- Plusieurs activités collaboratives.
- Communication sécurisée : les informations internes entre partenaires restent confidentielles.

Dans les processus métiers collaboratifs [Wes19], l'objectif de la modélisation et de la mise en oeuvre de processus n'est pas seulement l'efficacité, mais aussi la traçabilité exacte de ce qui a été réellement fait. Cet aspect est également présent dans la gestion des expériences scientifiques, où le lignage des données est un objectif important du support de processus. Plusieurs défis existent dans le domaine des processus métiers collaboratifs, l'une vient de l'autonomie des organisations et d'un manque de confiance, par exemple, quel partenaire exécute une tâche donnée, à quelles données chaque participant peut accéder.

La blockchain fournit une plate-forme appropriée pour exécuter des processus collaboratifs de manière fiable [Pin20]. Dans l'ensemble, les blockchains offrent une plateforme transparente et infalsifiable qui permet aux parties de suivre les actions des autres.

1.4.2 Exécution des processus métiers collaboratifs avec la blockchain

Dans cette section nous présentons les travaux antérieurs concernent les processus métiers collaboratifs de chorégraphie ou de collaboration avec la blockchain.

1.4.2.1 Collaborations de processus métier

Le premier travail qui a étudié le problème du manque de confiance dans les processus métier collaboratif en utilisant la blockchain a été rapporté dans [IW16]. Les auteurs ont développé une technique pour exécuter les processus métiers collaboratifs sur la blockchain Ethereum. Leur système est basé sur un composant principal nommé traducteur prenant en entrée une spécification d'un modèle de processus métier et génère des smart contracts selon les règles de traduction. A partir de ce modèle des instances des smart contracts sont générés et peuvent être implémentés sur la blockchain pour surveiller l'état d'exécution de processus et les échanges des messages entre tous les participants.
Cependant pour interagir avec le monde extérieur ce technique consiste à utiliser des déclencheurs qui reçoivent des appels d'API de composants externes et mettent à jour l'état du processus dans la blockchain en fonction des observations externes. Bien qu'ils montrent correctement que la blockchain peut éliminer l'exigence envers le tiers de confiance, mais leur solution ne résout pas complètement le problème de confidentialité des réseaux blockchain et entraîne un coût important pour le déploiement des smart contracts.

Une évaluation de [IW16] a mis en évidence par Garcia et al.[LGBW17] pour exécuter des processus métiers en blockchain avec une optimisation dans l'utilisation des ressources. Cette approche est apparu dans le but de minimiser le nombre des créations des contrats, la taille du code, les données des smart contracts et la fréquence des écritures des données. Les auteurs de cet article ont proposé une méthode optimisée qui permet de traduire un modèle BPMN en Petri-net minimisé et le compiler. Puis pour éviter le coût de déploiement de chaque instance de contrat ils ont proposé un seul smart contract qui peut gérer plusieurs instances en parallèle. Cela peut minimiser la consommation de gaz en codant l'état actuel du modèle de processus en tant que structure des données optimisée et réduire le nombre des opérations nécessaires pour exécuter une étape de processus mais aussi présente quelques limites au niveau du contrôle d'accès et l'encodage de flux de contrôle.

En comparaison avec les approches précédentes Caterpillar [OLP19] suit une approche différente où les échanges entre les processus ne sont pas exécutés via les échanges des messages mais à travers la blockchain. Cette proposition est considérée comme le premier moteur d'exécution de processus basé sur la technologie de blockchain et capable de gérer des modèles de processus avec des sous-processus. Caterpillar a modélisé le processus métier collaboratif de la même manière qu'un BP intra-organisationnel exécuté au-dessus d'un BPMS traditionnel, en effet, il est modélisé comme une orchestration. Les processus métier collaboratif prennent en charge les sous-processus où une instance d'un processus peut être liée à d'autres instances qui sont également enregistrées sur Blockchain.
Ce système fait soutenir l'exécution de processus métier collaboratif avec toutes les capacités de confiance de la technologie de la blockchain mais elle n'implémente aucun mécanisme de contrôle d'accès, ce qui signifie que n'importe quelle partie peut modifier l'état d'exécution de toute instance de processus.

1.4.2.2 Chorégraphies de processus métier

Une approche alternative pour vérifier la bonne exécution des instances de chorégraphie de processus métier au-dessus de la blockchain Bitcoin via des jetons spécialisés a été présenté par Prybila et al.[CP17]. En effet Ils ont mis certaines hypothèses où le participant qui démarre l'exécution remet le contrôle via un jeton à un premier partenaire approprié pour l'exécution d'une activité de processus spécifique. Ainsi, le partenaire sélectionné passe l'exécution à un autre participant à la chorégraphie pour effectuer la tâche suivante et le jeton de contrôle stocke l'état d'exécution du processus. Les participants peuvent documenter l'avancement de ce dernier et le transfert à d'autres participants en soumettant des nouvelles transactions qui propagent le jeton. Cette méthode augmente la confiance en offrant une grande flexibilité, mais ne tient pas compte de la confidentialité des données.

L'architecture de blockchain peut fournir les composants cruciaux nécessaires pour finalement faciliter la mise en oeuvre des processus inter-organisationnels sécurisés et sans confiance. Ceci est souligné par Mendling et al. dans leur discussion sur les opportunités et les défis du BPM avec les blockchains [JM18]. Ils affirment que tout les domaines des chorégraphies peut être revitalisé par la technologie la blockchain. Bien que les diagrammes de chorégraphie BPMN 2.0 n'aient pas trouvé une large adoption dans l'industrie, Mendling et al. les considèrent toujours comme l'une des approches prometteuses pour la conception de processus métier inter-organisationnels

Brahem et al. [AB19] ont utilisé la technologie de blockchain pour exécuter la chorégraphie de processus métier d'une manière qui respecte l'aspect collaboratif et résout le problème de la confiance. Cette approche est inspirée de la proposition de Caterpillar[4] en prenant en charge la chorégraphie de processus. Les auteurs ont proposé des règles de transformation pour reconstruire des annotations déclaratives XML en smart contract écrit dans un langage de programmation Solidity. La transformation est représentée par une production de chorégraphie à partir de XML puis d'exécuter ce processus en smart contract.

Ladleif et all. étendre les chorégraphies BPMN pour fournir une sémantique opérationnelle qui se repose sur les capacités de la blockchain [JL19]. Les auteurs proposent une approche qui favorise les exécutions des échanges des messages entre participants (typique des chorégraphies) dans la blockchain. Ils considèrent la blockchain comme une infrastructure d'exécution partagée. Ainsi qu'ils étendent les diagrammes de chorégraphie BPMN avec des objets des données pour représenter des informations publiques sur la blockchain, avec des sous-chorégraphies pour limiter la visibilité des données, avec des passerelles contrôlées par smart contract et avec une sémantique axée sur les transactions.

1.5 Positionnement des approches existantes

La technologie de la blockchain est également un véritable moyen pour automatiser les processus distribués grâce aux smart contracts, les transactions sont plus sécurisées, la transparence et l'accessibilité sont améliorées et l'intégrité des données est garantie. Ces atouts majeurs sont les véritables facilitateurs de l'automatisation des processus métiers où il n'est plus besoin de confirmation des transactions ni des étapes intermédiaires.
En effet les travaux de recherche que nous avons étudiés se concentrent que sur l'automatisation des processus métiers en blockchain. Seulement la contrôlabilité et la traçabilité des activités métiers ont été améliorées, aucune des études ne tient compte au cas d'échec des tâches et de comportement du processus métier dans de tels cas.

Conclusion

Dans ce chapitre nous avons présenté un contexte théorique sur la technologie de blockchain et le concept de processus métier. Ainsi que les approches qui sont positionnées sur la combinaison de ces technologies, bien que ces propositions assurent un bon niveau de contrôlabilité et de confiance sauf qu'il ne permettent pas le traitement des échecs.
Cette étude nous a conduit a conclure au besoin de modèle de transaction avancée (MTA) qui est responsable sur l'exécution des activités de manière syntaxique. Nous avons mené a ce effet une étude sur les processus métiers transactionnels dans le chapitre suivant.

Chapitre 2

Processus métiers transactionnels

2.1 Introduction

Les processus métier sont très utilisables dans les petites et grandes organisations. Les modèles de transaction avancés (ATM) se concentrent sur le maintien de la cohérence des données et ont fourni des solutions à de nombreux problèmes tels que l'exactitude, la cohérence et la fiabilité dans les environnements de traitement des transactions et de gestion de bases de données. Une évolution importante dans ce contexte est de combiner le concept transactionnel de modèle de transaction avec le processus métier.
Dans ce chapitre nous présentons les modèles de transaction avancées en expliquant le modèle de transaction avec un aperçu sur les modèles des transactions flexibles et on se termine avec les processus métiers transactionnels et les approches antérieurs.

2.2 Modèles de transaction Avancés (MTA)

Les modèles de transaction avancés (MTA) peuvent être considérés comme diverses extensions de transaction traditionnelle qui assouplissent une ou plusieurs contraintes de modèle ACID pour répondre aux exigences spécifiques des applications complexes, garantir une bonne exécution d'un ensemble des tâches et de répondre aux besoins de correction et de fiabilité [Bhi05, TW08].
En effet le modèle de transaction traditionnel avec ses propriétés ACID soit très simple et sécurisé, il n'a pas la capacité de prendre en charge des applications avec des transactions de longue durée et / ou complexes ce qui nécessitent une atomicité et une isolation relâchées [AC98, TW08]. Les MTA, tels que Sagas, le modèle de transaction imbriquée, le modèle de transaction à plusieurs niveaux et le modèle de transaction flexible, ont été définis pour mieux prendre en charge les transactions de long-durée dans un environnement distribué. Ces modèles de transaction avancés ainsi que d'autres modèles décrits dans [Elm92] peuvent être classés en fonction de diverses caractéristiques qui incluent la structure de transaction, la concurrence intra-transaction, les dépendances d'exécution, la visibilité, la durabilité, les exigences d'isolation et l'atomicité des échecs et ils permettent de regrouper leurs opérations en structures hiérarchiques [JS97].

L'intérêt des modèles de transaction avancés se concentrent sur le maintien de la cohérence des données et ont fourni des solutions à des nombreux problèmes tels que l'exactitude, la cohérence et la fiabilité dans les environnements de traitement des transactions et de gestion de bases des données [WS97].

2.2.1 L'origine de MTA

L'approche transactionnelle est apparue initialement dans le contexte des systèmes de gestion de bases de données avec des modèles ACID.
Le modèle des transactions ACID désigne une séquence d' opérations de lecture ou d'écriture sur un ensemble d'objets comme une unité de traitement, appelée transaction qui vérifie les propriétés suivantes [BM09] :

- Atomicité : Fait référence à une propriété tout ou rien, soit une transaction se termine avec succès (la transaction est validé), soit la transaction est interrompue (elle est abandonnée).
- Consistance : Les transactions fonctionnent toujours sur une vue cohérente de la base de données et laissent la base des données dans un état cohérent.
- Isolation : les effets d'une transaction en cours d'exécution sont invisibles aux transactions concurrentes
- Durabilité : Les effets d'une transaction validée sont persistants et ne peuvent plus être remis par une défaillance, ni par une autre transaction.

les modèles ACID sont produits pour prendre en charge les applications qui nécessitent des structures de contrôle plus complexe qu'une simple séquence d'opérations de lecture / écriture et une exécution de longue durée [HGM87, SB11].

En effet, l'expressivité du modèle de transaction traditionnel a conduit au développement de nombreux nouveaux modèles de transaction, dits étendus. Dés lors, Ces modèles qui sont les MTAs tentent de surmonter les limites de modèle traditionnel (ACID), en introduisant la structure interne d'une transaction ou en relâchant certaines des propriétés ACID généralement l'atomicité et l'isolation[AC98].
A cet égard, l'atomicité relâchée est signifie que chaque application peut avoir sa propre atomicité de défaillance dépendante de l'application. Un flux de travail peut survivre et progresser même si certaines de ses tâches ne se terminent pas correctement [JE95]. Et l'isolation relâchée est signifie que les activités peuvent externaliser des résultats non engagés et libérer des ressources pour atteindre un degré plus élevé de concurrence. Cette caractéristique est complétée par le concept de compensation qui permet une annulation sémantique des activités déjà engagées [JE95].

Les MTA ont été étendu par la définition de ce modèle en relâchant le propriété d'atomicité pour définir la place de « Atomicité d'échec » ou échec partiel d'une activité de processus et la propriété d'isolation pour permettre gérer la concurrence inter-processus comme dans le modèle de transaction imbriquée ouvert (Open Nested Transaction Model) [GW92] et la compensation en tant que récupération en arrière dans le modèle Saga [HGM87]. Le modèle de transaction flexible ensuite où se concentre notre travail a été inspiré de modèle de transaction imbriquée pour introduire la notion de cohérence.

La logique fondamentale des ATMs [TW08] est de diviser une transaction en sous-transactions selon la sémantique des applications. Ces sous-transactions, également appelées transactions des composants, peuvent à nouveau être divisées si nécessaire. Les transactions avancées peuvent effectuer des tâches plus complexes et plus durables.

2.2.2 Modèle de transaction flexible

Depuis sa proposition, le modèle des transactions flexibles (FTM) a subi plusieurs changement. Au départ, Les transactions flexible[Bhi05, JS97, MR94, AZ94] ont été proposées comme modèle de transaction adapté à un environnement multi-bases de données. Une transaction flexible [WS97] est un ensemble de tâches avec un ensemble de sous-transactions fonctionnellement équivalentes pour chacune il peut spécifié un ensemble des dépendances d'exécution, y compris les dépendances d'échec, les dépendances des succès ou les dépendances externes. Pour assouplir les exigences d'isolation, les transactions flexibles utilisent la compensation et les exigences d'atomicité globales est assouplissent spécifiant les états acceptables pour la fin de la transaction flexible dans laquelle certaines sous-transactions peuvent être abandonnées.

Selon [WWJ93] une transaction flexible est spécifiée en fournissant : la pré-condition de la transaction globale, un ensemble de sous-transactions, les états visibles de l'extérieur de chaque sous-transaction et les transitions possibles entre ces états visibles de l'extérieur, les pré-condition et post-condition pour les transitions possibles de chaque sous-transaction, et la post-condition de la transaction globale.
Ainsi, une transaction flexible comprend un protocole de coordination le contrôle et le flux de données entre les transactions de composants. Ce protocole est spécifié par une machine à états. La pré-condition à la transaction globale spécifie ses états initiaux et la post-condition de la transaction globale spécifie l'ensemble des conditions définissant ses états finaux (les états acceptables) ; certains de ces états finaux peuvent être considérés comme la réussite de la transaction globale ou l'échec dans un état cohérent de la transaction globale.
L'ensemble des états visibles de l'extérieur d'une sous-transaction comprend généralement : initial, en cours d'exécution, validé, prêt à valider, abandonné. Les pré-conditions et post-conditions peuvent être énoncées en termes de variables d'état pour les sous-transactions et de variables supplémentaires.

En d'autre terme une transaction flexible [AZ94] en tant que transaction de premier niveau est un ensemble des tâches, chacune ayant un ensemble de sous-transactions fonctionnellement équivalentes entre lesquelles deux ordres, un ordre de priorité et un ordre de préférence. chaque transaction globale a au plus une sous-transaction et chacune de ces sous-transactions peut être classée par l'une des propriétés de terminaison (compensable, rejouable ou pivot).
Suivant [AZ94] le modèle de transaction flexible prend en charge le flux de contrôle d'exécution flexible en spécifiant deux types des dépendances parmi les sous-transactions d'une transaction globale :

- Dépendances d'ordre d'exécution entre deux sous-transactions.
- Dépendances alternatives entre deux sous-ensembles de sous-transactions.

Cependant il présente deux types des relations de flux de contrôle pour définir sur ces sous-ensembles :

- La relation de priorité pour définit les dépendances correctes de classement d'exécution parallèle et séquentielle parmi les sous-transactions.
- La relation de préférence définit les dépendances de priorité entre les ensembles alternatifs de sous-transactions pour la sélection lors de l'exécution.

Propriétés transactionnelles

[HSS02] adopte les sémantiques transactionnelles de tâche rejouable, compensable et pivot.

- Compensable : L'impact produit par une activité **compensable** peut être sémantiquement annulés par l'exécution d'une activité de compensation, cela constitue une garantie de compensabilité de l'activité.
- Rejouable : Une activité est dite **rejouable** lorsqu'elle offre la garantie de se terminer avec succès après un nombre fini d'activation.
- Pivot : Une activité est dite **pivot** si elle se termine avec succès et ses effet ne peuvent pas être compensé ou elle échoue et compense le travail déjà effectué.

[HSS02] utilisent ces propriétés transactionnelles pour définir des transactions flexibles et correctes. Ces transactions flexibles doit respecter certaines règles :

- Toute tâche entre deux activités pivots ou avant la première activité pivot doit être compensable.
- Après une activité pivot, il peut y avoir plusieurs branches alternatives, avec priorité parmi elles.
- La dernière branche alternative après une activité pivot doit être sure de se terminer. Cela signifie que toutes les tâches de la dernière branche alternative doivent être rejouables.

La figure 2.1 présente d'un exemple de transaction flexible. Les activités A,B et E sont compensables. Les activités C,D et F sont pivots. Les activités G,H et I sont rejouables. Une transaction est recouvrable en arrière si son premier pivot C n'est pas exécuté puisque A et B sont compensables. Après l'exécution de C il y a trois alternatives dont la dernière est sûrement se terminer puisque H et I sont rejouables. De même la transaction est recouvrable jusqu'à C si les des deux autres alternatives échouent. Enfin, après l'exécution de son deuxième pivot F, la transaction est sûre de se terminer puisque G est sûre de se terminer.

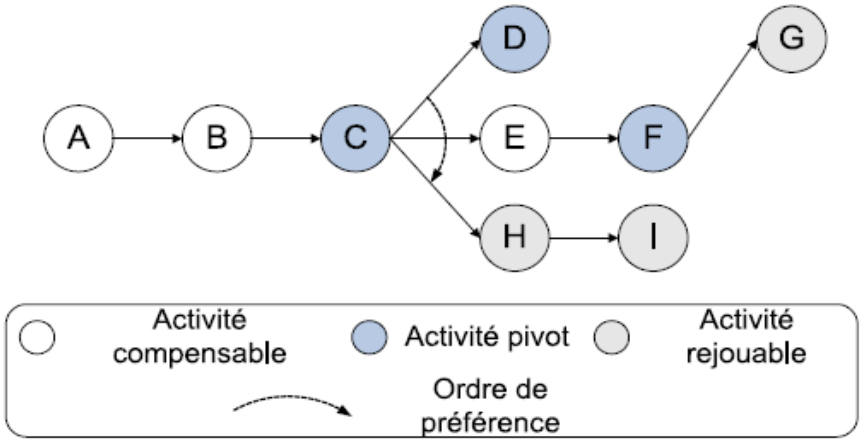


FIGURE 2.1 – Un exemple de transaction flexible [Bhi05]

Alternative - semi-atomicité

Les travaux sur les transactions flexibles [JS97, AZ94] discutent le rôle des transactions alternatives qui peuvent être exécutées sans sacrifier l'atomicité de la transaction globale. En effet, la semi-atomicité [MG12] a été proposé pour les transactions flexible pour résoudre l'imposition du modèle Saga dans lequel le processus se termine avec succès ou il est compensé et son idée principale consiste a étendre le modèle de la saga en fournissant des chemins d'exécution alternatives afin d'assurer des exécutions correctes pour certains sous-transactions même en présence d'échecs [Bhi05, MD05]. Elle ajoute aux avantages des sagas la possibilité de ne compenser que des parties d'un processus et d'utiliser ces chemins d'exécution alternatifs pour éviter d'avoir abandonner le travail déjà effectué si il y a des défaillances.
En d'autres termes la transaction globale de la transaction flexible est divisée en un ensemble des sous transaction équivalente, une fois qu'une sous transaction ne peut pas exécuter sa équivalente sera lancé. Ces sous transactions sont représentées dans un ordre partiel représentatif sont liées par la relation de précedence.

compensabilité

Les transactions dans l'environnement multi-base des données peuvent durer longtemps [JS97, YL92]. Ce qui provoque des problèmes surviennent lors de l'application d'une isolation stricte dans des applications de longue durée.
Par conséquent, la granularité d'isolation de la transaction globale doit être réduite. Chaque sous-transaction est associé à une sous-transaction de compensation qui peut annuler sémantiquement les effets d'une sous-transaction engagée si nécessaire.
L'annulation d'une transaction incomplète (ou la récupération en arrière) est un mécanisme de réparation accepté pour les transactions annulées. Nous pouvons définir une tâche de compensation ou une série de tâches par les tâches qui peuvent efficacement annuler ou réparer les dommages causés par une tâche ayant échoué dans un processus.

2.3 Processus métier transactionnel

Étant donné que les processus métiers contiennent des activités qui accèdent à des ressources des données partagées et persistantes, ils doivent être soumis à une sémantique transactionnelle [WG09]. Le terme de processus métier transactionnel (TBP) a été introduit dans le but de spécifier la pertinence des transactions dans le processus métier afin d'assurer un certain degré d'exactitude et de fiabilité [Bhi05].

En particulier, Les processus métier transactionnels [WG09, JS97] sont apparus comme une combinaison entre un processus métier et un modèle de transaction avancé. Ils visent a assurer une exécution fiable et correcte même en présence de concurrence et d'échec. Néanmoins, ils partagent les objectifs de certains des ATMs en termes de pouvoir appliquer une sémantique de transaction assouplie à un ensemble des activités.
Dans un TBP les tâches sont mappées aux transactions constitutives d'une transaction avancée prise en charge par un MTA et le flux de contrôle est défini comme des dépendances entre étapes transactionnelles. Ainsi il peut fournir des propriétés transactionnelles pour prendre en charge la récupération, et utiliser la sémantique du système et de l'application pour prendre en charge l'exécution correcte d'une application multi-système.

2.3.1 Flux de contrôle

Le flux de contrôle [WG09] décrit l'ordre dans lequel les activités de processus sont activées ainsi que l'ordre de préférence d'exécution des activités parmi les alternatives. Le flux de contrôle d'un processus métier spécifie l'ordre partiel des activations d'activité des composants. Dans une instance de processus métier où toutes les activités sont exécutées sans échec ou annulation, il définit les dépendances d'activité.

2.3.2 Comportement transactionnel

Le comportement transactionnel [WG09] est observé en cas de défaillance d'une activité, il définit des mécanismes de récupération prenant en charge la gestion des pannes pendant l'exécution. Ces mécanismes de récupération sont utilisés pour garantir la tolérance aux pannes de processus afin de répondre de manière approprié aux situations des échecs attendus (également appelés exceptions). Fondamentalement, le comportement transactionnel est décrit à travers les propriétés transactionnelles de l'activité et le flux transactionnel représentant respectivement les dépendances transactionnelles intra et inter-activités après les échecs d'activité.
Les dépendances transactionnelles jouent un rôle important dans la coordination et l'exécution d'une instance de processus métier. Ces dépendances définissent les relations qui peuvent exister entre l'état d'exécution de l'activité de rapport des événements (annulé, échoué et terminé) d'une ou deux activités spécifiant respectivement le les dépendances d'état intra-activité (les propriétés transactionnelles d'activité) ou les dépendances d'état d'inter-activité (le flux transactionnel).

Propriété transactionnelle d'une activité Chaque activité peut être associée à un état du cycle de vie qui modélise les états possibles par lesquels les exécutions de cette activité peuvent passer, et les transitions possibles entre ces états. Ces transitions décrivent les dépendances d'état intra-activité dont dépendent des propriétés transactionnelles des activités.
Les principales propriétés transactionnelles que nous considérons sont réjouables et pivotantes et compensables.

Flux transactionnel Le flux transactionnelle [Bhi05, WG09] engloble l'ensemble des propriétés transactionnelles d'une activité de processus métier qui sont l'équivalent des propriétés de terminaison dans le FTM (compensables, pivot et rejouable).
Il décrit également des dépendances d'état inter-activités après l'échec d'une activité qui permettent à un BP de réussir ou d'échouer les dépendances alternative, les dépendances de compensation et les dépendances d'annulation.
En effet après l'échec d'une activité, le flux transactionnelle essaie d'abord d'exécuter une alternative (si elle existe) pour activer une autre activité afin de reprendre l'exécution de l'instance, sinon (il n'y a une alternative), en cas d'échec globale de processus métier on compense le travail déjà effectué, et annule toutes les exécutions en cours en parallèle.

Le comportement transactionnelle(les propriétés et les dépendances transactionnelles) est dépend de flux de contrôle(la logique d'exécution du processus métier). En effet pour garantir une exécution fiable la spécification d'un mécanisme de récupération défini par le comportement transactionnel doit respecter des règles qui dépendent partiellement de flux de contrôle.

2.3.3 Mécanisme de recouvrement

Les mécanismes de récupération spécifiés par les dépendances transactionnelles permettent aux instances de processus métier échoués de revenir à un état cohérent.
C'est grâce au mécanisme de récupération l'état de défaillance incohérent peut être corrigé et l'exécution peut être redémarrée à partir d'un point cohérent [WG09].

Mécanisme de récupération en arrière (compensation)

La notion de récupération [WG09, Bhi05, JS97] est importante dans les systèmes de processus métier, l'annulation des transactions incomplètes ou la récupération en arrière est un mécanisme de réparation accepté pour les transactions annulées.
L'exécution de l'instance reprend à partir d'un état cohérent situé avant l'activité qui a échoué. Il peut également être nécessaire de démarrer une activité de compensation qui supprime les effets secondaires incohérents et annule sémantiquement l'effet de l'activité échouée correspondante. Une récupération en arrière peut être effectuée en appliquant successivement des activités de compensation.
Prenons l'exemple de la figure 2.2 pour définir le mécanisme de récupération en arrière pour faire face à l'échec de l'activité A. La dépendance de compensation de B vers A indique que B est componsé quand l'activité A échoue.

Mécanisme de récupération en avant (Alternatives)

Les dépendances d'alternative [WG09, HSS02] permettent d'exprimer des alternatives d'exécution comme mécanismes de récupération en avant. Une fois que le pivot principal s'est terminé avec sucées, le processus sera dans un état terminé et puisqu'il a une alternative finale qui ne contient que des activités réjouables, un processus dans cet état est garanti. Dans le cas de récupération en avant l'activité qui est située après l'activité qui est échoué ou dans un autre chemin, est exécutée pour terminer correctement l'exécution de processus métier sans effets secondaires.
Prenons l'exemple de la figure 2.2

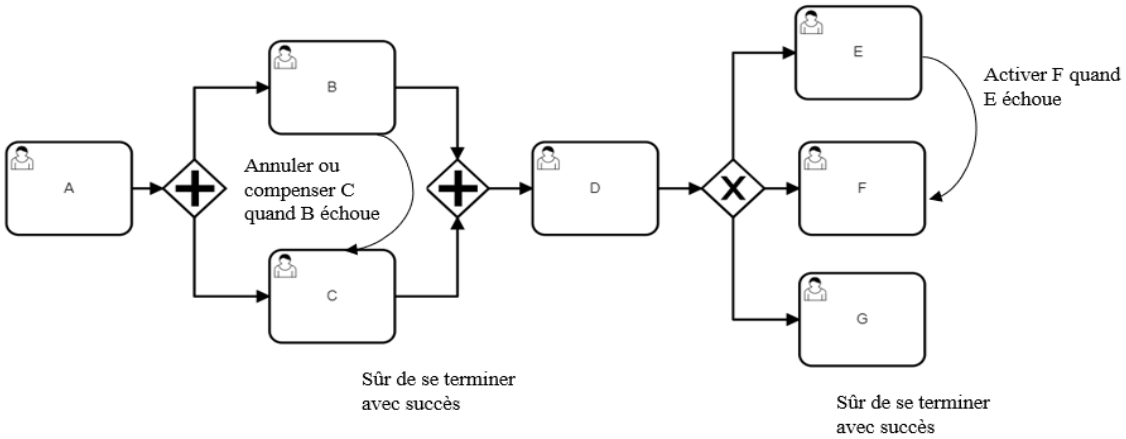


FIGURE 2.2 – Un exemple de processus métier avec les mécanismes de recouvrement

Une dépendance d'alternative de E vers F indiquant que le service F est activé quand le service E échoue. Cette dépendance permet de spécifier F comme une alternative de livraison pour E permettant ainsi de définir un mécanisme de récupération en avant en cas d'échec de F.

2.4 Exécution des processus métiers transactionnels

Afin de garantir une exécution fiable, un processus métier doit être en mesure pour garantir les propriétés transactionnelles qu'il contrôle. Un système de gestion fiable doit maintenir la cohérence des données pour les instances de processus en cas d'échec survenu lors de leur exécution. Depuis longtemps l'aspect transactionnel n'est pas utilisé dans les travaux réalisés, dans la suite nous citons quelques travaux d'exécution des processus métiers transactionnels pour les services web.

Les auteurs de [SB11] ont proposé un modèle de service Web composite transactionnel qui intègre la flexibilité du workflow et la fiabilité des modèles transactionnels avancés en se basant sur des techniques selon la spécification de besoin en termes de flux de contrôle et d'atomicité de défaillance.
Le modèle proposé est basé l'état du service (par exemple. Initial, abandonné, actif, terminé) et la transition (par exemple. Abandonné, activé, annulé, échoué, terminé) qui déterminent les états possibles par lesquels une exécution de service peut passer, et le transitions possibles entre ces états. Le modèle de processus métier est défini par les dépendances entre les services. Ces dépendances sont représentées par des conditions sur l'état du service (par exemple, l'état terminé du service permet d'activer la transition d'un autre service).

Une approche alternative pour la modélisation et la mise en oeuvre de processus métier proposé dans [JEF12]. C'est une méthodes classiques de conception et de mise en oeuvre des processus métiers basés sur des services Internet qui utilisent des approches orientées processus ont des difficultés avec la gestion des exceptions, y compris les annulations et les échecs. Les auteurs de cet article décrive une approche WED-flow qui permet de fournir une récupération transactionnelle via une évolution incrémentielle de la gestion des exceptions au moment de l'exécution qui peut être appliquée dans les applications de processus métier qui utilisent les services Internet pour un objectif principal est de réduire la complexité de la gestion des exceptions. WED-flow combine les concepts de modèles de transaction avancés, d'événements et d'états de données. A la suite il est capable de garantir des contraintes d'intégrité à l'échelle de l'application grâce aux mécanismes de récupération en avant et en arrière.

2.5 Positionnement des approches existantes

L'aspect transactionnel est également un véritable moyen pour garantir une exécution correcte des processus métiers. L'utilisation des modèles de transaction avancés est important pour qu'on puisse exécuter des application complexe et maintien la cohérence des données. En effet les travaux de recherche que nous avons étudiés se concentrent sur l'utilisation de l'aspect transactionnel pour assurer une exécution fiable des modèles de service web, sauf que ils sont limité au modèle d'orchestration qui ne subit pas d'une interaction entre plusieurs participants.

2.6 Conclusion

Dans ce chapitre, nous avons présenté la combinaison entre les modèles de transaction avancée et les processus métiers pour répondre au besoins de correction. Bien que ces systèmes permet de gérer la gestion d'échec du coup nous l'avons utilisé dans notre approche qui se base sur l'exécution des processus métiers transactionnelle sur la blockchain en utilisant le smart contract.

Chapitre 3

Approche proposée

Introduction

L'intégration de processus métier en blockchain peut augmenter la performance d'une entreprise en termes de confiance, transparence intégrité des données et sécurité. Malgré ces bénéfices certaines des faiblesses apparentes aux processus métier comprennent les critères d'exactitude et manque de support adéquat pour la fiabilité en présence de pannes et d'exceptions doivent être abordées.
Ce travail vise a créer une solution pour résoudre les problèmes de manque de fiabilité dans l'exécution des processus métiers sur la blockchain et dépasse les limites des recherches existantes. Pour atteindre cet objectif et garantir l'exactitude des transactions nous proposons une approche qui s'appuie sur les smart contract et les étendes à la mise en oeuvre des processus métiers transactionnels.
Dans ce chapitre nous présentons l'approche proposée en deux parties. Une partie pour présenter l'exécution de processus métier transactionnel en smart contract de la blockchain et la deuxième partie présente une étude théorique sur le cas d'exécution de processus métier transactionnels collaboratifs sur une blockchain avec permission.

Partie1 : Smart contract basé sur le processus métier transactionnel

Les smart contracts prennent de l'ampleur en tant que technologie appropriée pour garantir une exécution fiable des processus métiers et des collaborations BPMN dans un environnement ouvert. Dans notre approche nous avons appui sur un système BPMN existant qui est développé au-dessus de la plateforme Ethereum appelé Caterpillar [OLP19]. En effet, nous avons étendu Caterpillar dans le cas de processus métier transactionnel ce qui nous donne un avantage à notre travail pour assurer une bonne exécution qui réagit aux échecs de modèle BPMN dans la blockchain Ethereum.
Nous utilisons la technologie des smart contracts pour permettre l'exécution fiable et décentralisée de processus métier dans un environnement ouvert et par les processus métier transactionnels pour gérer les échecs et garantir l'exactitude transactionnelle des exécutions BP.

3.1 Aperçu de l'architecture de Caterpillar

La figure 3.1 illustre l'architecture qu'on a étendu. Ce travail vise à prendre une spécification BPMN 2.0 comme entrée et génère le flux de contrôle correspondant dans un smart contract (SC).

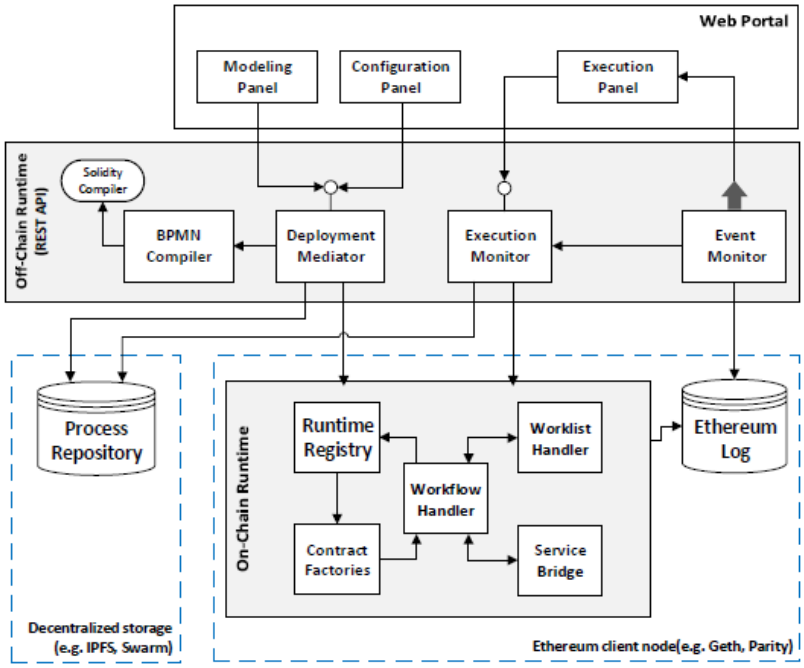


FIGURE 3.1 – Architecture de Caterpillar [OLP19]

L'architecture est compris trois blocs principaux en plus de la partie de stockage : Une partie de Web Portal présente une interface utilisateur qui est composée d'un panneau de modélisation pour créer et soumettre le modèle BPMN, un panneau de configuration pour afficher tous les smart contracts générés à partir des modèles BPMN soumis précédemment stockés dans une base de données MongoDB et déployer une instance d'un smart contract et un panneau d'exécution pour exécuter l'exécution de l'un des processus déployés sur la blockchain.

La partie de back-end off-chain de l'application est composé de : un module BPMN-to-Solidity Converter qui collecte les informations de la modélisation panneau et stocke les smart contracts de processus résultants dans le stockage ; un contrôleur composé d'un médiateur de configuration et d'un moniteur d'exécution pour communiquer avec la blockchain et l'interface utilisateur via une API REST.

Et la partie de Blockchain (Caterpillar Runtime)de backend on-chain consiste en plusieurs smart contracts (registre et usine de contrat) pour gérer et surveiller le déploiement des smart contracts de processus. Deux types de contrats de processus sont définis dans en fonction de la nature de la tâche BPMN : contrat de gestionnaires de liste de travail (Worklist) pour la mise en oeuvre des tâches utilisateur (tâches qui nécessitent une action de l'utilisateur) et gestionnaires de services (Service task) pour les tâches de service basées sur des oracles.

Dans notre approche nous intéressons à la partie backend off-chain où nous modifions les templates bpmn2sol et worklist2sol.

Backend off-chain

La partie de off-chain contient trois modules principaux compilateur BPMN, un médiateur de déploiement et un moniteur d'exécution.

Compilateur BPMN

Le compilateur BPMN permet de compiler le modèle de processus BPMN en smart contract selon deux étapes :

Dans la première étape, le modèle de processus est compilé en smart contract solidity. En premier étape, le modèle de processus BPMN est compilé en un ensemble de smart contract Solidity avec des méta-données supplémentaires, appelées dictionnaire de compilation. Ce dictionnaire de compilation présente une structure des données qui comprend des informations permettant de mapper les éléments de modèle BPMN au code généré. Ces informations incluent le nom de la méthode contractuelle associée à toute activité, un index entier unique attribué à chaque élément, ainsi que le type d'élément respectif.

En deuxième étape de processus de compilation, Caterpillar met en place l'ensemble des smart contracts produits dans la première étape et l'ensemble des smart contracts déjà existants. Ils sont transmis au compilateur solidity qui produit le bytecode EVM et les définitions ABI qui sont utilisées pour déployer ces smart contracts sur Ethereum.
La figure 3.2 suivante représente les étapes de compilation.

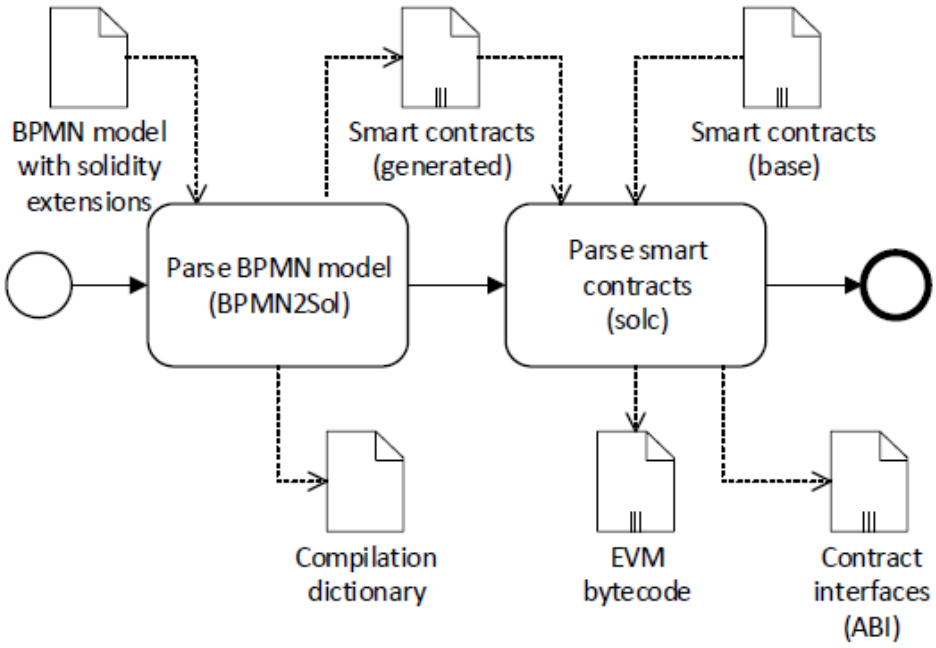


Figure 3.2 – Compilateur BPMN [OLP19]

Le compilateur Caterpillar produit trois smart contracts à partir d'un modèle d'entrée. Le premier implémente les perspectives de flux de données et de contrôle ; la perspective des données est intégrée dans le cadre de l'implémentation du flux de contrôle. Le deuxième contrat nommé Worklist pour gérer l'exécution des éléments de travail par les parties prenantes et pour server à envoyer / recevoir les données de processus.
Le troisième contrat nommé factory fournit un mécanisme par défaut pour créer des instances du processus. Certains éléments de modélisation tels que les activités multi-instances et les activités d'appel sont mises en oeuvre dans des contrats séparés.

Déploiement des smart contracts

Le moment où le processus de compilation se termine et les résultats stockés dans le référentiel, le contrat racine peut être instancié à partir du médiateur de déploiement. Les identifiants des contrats sont les hachages produits par le référentiel qui sert également de clé pour accéder aux méta-données de compilation.
La figure 3.3 montre les étapes effectuées par Caterpillar lors du déploiement d'un contrat avant de créer la première instance. Au départ, toutes les relations parent-enfant sont mises à jour dans le registre. Ensuite, pour tout contrat dans la hiérarchie des processus, les usines et les ressources correspondantes (les listes de travail et les services) sont instanciés et associés en conséquence dans le registre. Notez que l'instanciation d'un smart contract hors chaîne nécessite le bytecode produit par le compilateur solidity.
D'un autre part, l'appel ou l'invoke d'une fonction nécessite le contrat ABI en conjonction avec l'adresse d'une instance en cours d'exécution. Enfin, le processus racine est instancié puis démarré.

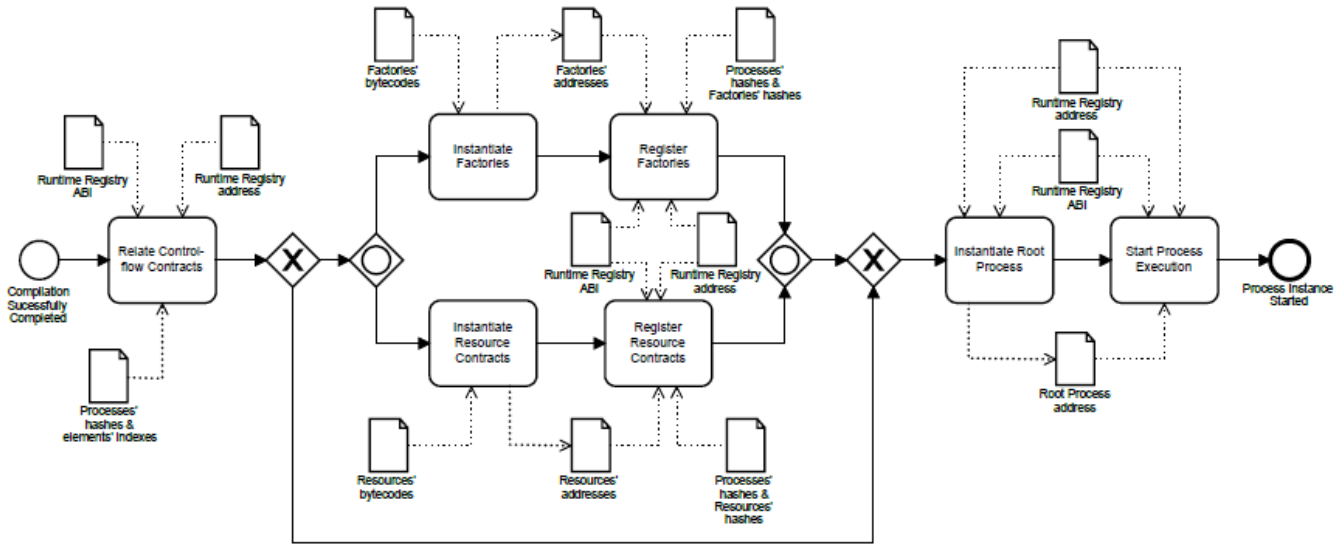


FIGURE 3.3 – Processus d'instanciation de Caterpillar [OLP19]

Interrogation de l'état du processus et exécution des tâches

Une fois qu'un modèle de processus a été déployé dans le runtime de CATERPILLAR pour exécution, toute instance de celui-ci évolue à partir de son état initial, exécutant un sous-ensemble des activités sous-jacentes jusqu'à ce qu'aucune activité ne soit considérée

comme candidate à l'exécution ou en cours d'exécution. Suivant cette intuition, la partie pertinente de l'état d'une instance de processus à un moment donné peut être identifiée avec le sous-ensemble d'activités qui sont dans l'état activé. Toute activité activée pendant l'exécution est automatiquement lancée par CATERPILLAR, simplifiant ainsi la tâche d'interrogation.

3.2 Solution proposée

Notre solution vise à évoluer le système de Caterpillar pour qu'il rende capable d'exécuter des processus métiers en prenant en compte le cas d'échecs des tâches et le comportement ultérieur des processus métiers dans de tels cas. Sur cette base nous proposons une approche qui s'appuie sur les smart contracts généré a partir de système Caterpillar et les étendons dans l'exécution du processus métier transactionnel.

En effet, nous nous essayons à mettre en oeuvre un TBP où son flux de contrôle qui est donné en entrée et son flux transactionnel qui suit les principes du modèle transactionnel flexible sont implémenté a la fois dans le smart contract généré. Nous avons également implémenté le flux transactionnel ainsi qu'un ensemble des mécanismes transactionnels tels que des propriétés transactionnelles qui sont affectées aux activités de processus et des états de terminaison qui nous appelons les états de terminaison transactionnelle pour examiner l'état d'échec de chaque activité en cours d'exécution. Ces derniers comprennent notamment des compensations comme backward recovery et des alternatives comme forward recovery qui sont exécutés dans le smart contract "Process SC".

D'une autre côté nous essayons de modifier "Oracle SC" en ajoutant une approbation pour les événements.

3.3 Spécification du flux de contrôle

La logique Caterpillar pour la mise en oeuvre du flux de contrôle repose sur la simulation du jeu de jeton et la définition du traitement à exécuter pour chaque état du cycle de vie d'une activité(activé, commencé, terminé), pour ce faire, Caterpillar utilise un générateur de code Solidity appelé «bpmn2sol template» qui est responsable sur la génération de code Solidity du «Process S.C.» à partir de la spécification de modèle BPMN.

En fonction de la position du jeton dans le processus, le modèle génère des fonctions prédéfinies à exécuter dans le «Process S.C.». En effet lorsque ce jeton est présent sur le bord entrant d'une activité appelée, le modèle BPMN-to-Solidity génère une première fonction nommée ActivityName_start(). Lorsque ce dernier est appelé, l'activité est considérée comme activée.

Ensuite, une deuxième fonction intitulée ActivityName() est générée ce qui signifie l'exécution de l'activité commence et l'activité est considérée comme commencée (enabled).

Enfin, le modèle produit une troisième fonction ActivityName_complete() signifiant que le jeton est consommé et qu'un nouveau est généré sur le bord sortant de l'activité, c'est-à-dire que l'exécution se termine et que l'état de l'activité est terminé.

Caterpillar utilise également un deuxième générateur de code Solidity qui est le «modèle Oracle-to-Solidity». Ce modèle produit le code de «Oracle S.C.». L'Oracle S.C.est associé au Process S.C.et gère l'interaction de ce dernier avec les utilisateurs et applications externes. lorsqu'une tâche utilisateur est lancée, il publie un événement Solidity dans le Ethereum log. De plus il peut également lire les données du Process S.C.pour les renvoyer à l'utilisateur externe.
Dans notre approche nous ajoutons quatre autres états de terminaison possibles qui sont abandonnés, annulés, échoués et compensés que nous appelons les états de terminaison transactionnelle. Nous discutons ces états dans la section suivante.

3.4 Spécification du flux transactionnel

Nous nous appuyons sur la logique de l'outil Caterpillar pour implémenter le flux transactionnel.
Tout d'abord, nous attribuons des propriétés transactionnelles aux activités de processus et enrichissons les cycles de vie des activités de processus avec des états transactionnels afin de modéliser leurs propriétés transactionnelles.
Ensuite, nous modifions les modèles de génération de code Solidity (modèles BPMN-to-Solidity et Oracle-to-Solidity) de manière à ce que, en fonction des propriétés transactionnelles de l'activité, les modèles génèrent des fonctions à exécuter pour chacun des états transactionnels.

3.4.1 Propriétés transactionnelles d'une activité

Pour chaque activité de processus BPMN nous ajoutons un attribut qui désigne la propriété transactionnelle. Les propriétés qui nous considérons dans notre approche sont **pivot**, **compensable** et **rejouable**. Chaque activité de processus peut avoir une ou plusieurs propriétés transactionnelles.
Le code de ci-dessous montre la sortie XML d'une tâche BPMN avec notre solution qui ajoute une propriété transactionnelle a une activité.

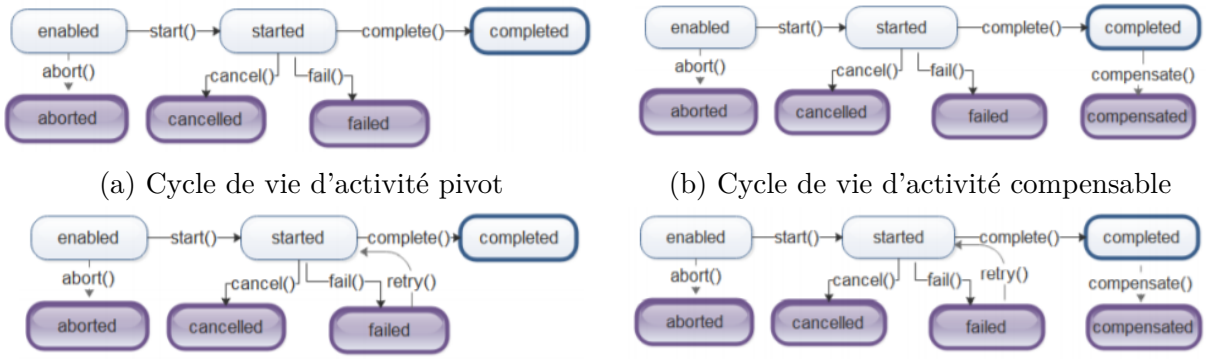
```
<bpmn:userTask id="Activity_01mh1q6" name="A">
 <bpmn:documentation> ():()-&gt;{ TransactionalProperty=compensatable;}
 </bpmn:documentation>
 <bpmn:incoming> Flow_0yc4snb </bpmn:incoming>
 <bpmn:outgoing> Flow_0yf21mt </bpmn:outgoing>
</bpmn:userTask>
```

Listing 3.1 – Sortie XML d'une tâche BPMN

La balise **<bpmn :userTask>** correspond à la description des informations générales sur la tâche.
La balise **<bpmn :documentation>** est l'origine destinée à fournir des informations supplémentaires sur la tâche, Caterpillar a utilisé pour stocker les valeurs saisies par l'utilisateur lors de l'exécution du processus sur la blockchain, et nous utilisons cette balise pour définir la propriété transactionnelle nécessaire pour l'activité de l'utilisateur.
Les deux dernières balises pointent vers les éléments BPMN connectés séquentiellement à la tâches.

3.4.2 Cycle de vie des activités

Chaque activité transactionnelle est associée à un diagramme à transition d'états qui modélise son comportement. Ce diagramme décrit les états possibles par lesquels l'activité de processus peut passer durant son cycle de vie pour chaque propriété transactionnelle. En effet Caterpillar a considéré dans son travail que l'état **terminé** pour la fin d'une activité, dans notre approche nous ajoutons des autres états de terminaison **abandonné, annulé, échoué, terminé** que nous appelons les états de terminaison transactionnelle. La figure 3.4 illustre le cycle de vie d'une activité en fonction des propriétés transactionnelles.



(a) Cycle de vie d'activité pivot

(b) Cycle de vie d'activité compensable

(c) Cycle de vie d'activité rejouable

(d) Cycle de vie d'activité compensable et rejouable

FIGURE 3.4 – Cycle de vie d'une activité en fonction des propriétés transactionnelles [Bhi05]

L'ensemble des états que nous avons considérés dans notre approche est **initial, abandonné, activé, annulé, échoué, terminé, compensé**.
Lorsqu'une activité est instanciée, elle est dans son état **initial**. Puis elle peut être **abandonné** avant d'être activé pour l'exécuter ou **activé** pour être exécuté. Une fois elle est activée, elle peut continuer son exécution ou peut être **annulé**. Si l'activité n'est pas annulée pendant l'exécution, elle peut atteindre son objectif et se terminer avec succès ou elle peut échouer. Une activité compensable peut être compensée après sa terminaison et passer ainsi à l'état compensé.

Une activité pivot : Le diagramme d'état d'une activité pivot est présenté dans la figure1.3 (a). Au départ elle est initialement à l'état **activé**. Ensuite, elle peut être soit **abandonné** (avant d'être activé pour l'exécuter) ou rien ne se passe et l'activité est lancée. Et lors de l'exécution, l'activité peut être **annulée**. Si l'activité n'est pas annulée durant l'exécution il existe deux possibilités soit l'activité se termine avec succès et atteint l'état **terminé** soit elle échoue et rencontre l'état à **échoué**.

Une activité compensable : Le diagramme d'état d'une activité compensable est présenté dans la figure1.2 (b), il désigne les mêmes transitions d'une activité pivot sauf que l'activité compensable pourrait être compensée après sa terminaison.

Une activité rejouable : Le diagramme d'état d'une activité réjouable est présenté dans La figure1.2 (c), il a les mêmes transitions d'une activité pivot sauf que l'activité réjouable peut être réactivée après l'échec.

Une activité compensable et réjouable : Le diagramme d'état d'une activité réjouable et compensable est présenté dans la figure figure1.2 (d), il a les mêmes transitions d'une activité pivot mais cette activité peut être réactivée après l'échec de plus elle peut être compensée après sa terminaison.

3.4.3 Implémentation de mécanismes transactionnels

Dans cette étape nous apportons quelques modifications aux modèles de génération de code Solidity. De cette manière, le «Process S.C» est enrichi pour mettre en oeuvre les mécanismes transactionnels sur le cycle de vie prolongé.

3.4.3.1 Définition de vecteur d'état

Pour chaque activité de processus BPMN nous définissons un vecteur d'état dont le but de capturer l'évolution de l'état de l'activité dans le smart contract de processus. Ce vecteur d'état est présenté sous forme d'un tableau de bits où chaque bit code un état différent qu'une activité peut atteindre. Un tableau de bits est codé sous la forme d'un entier non signé de 256 bits, qui est la taille de mot par défaut dans la machine virtuelle Ethereum. La figure 3.5 montre les vecteurs d'état associé a chaque état d'une activité. Dans notre

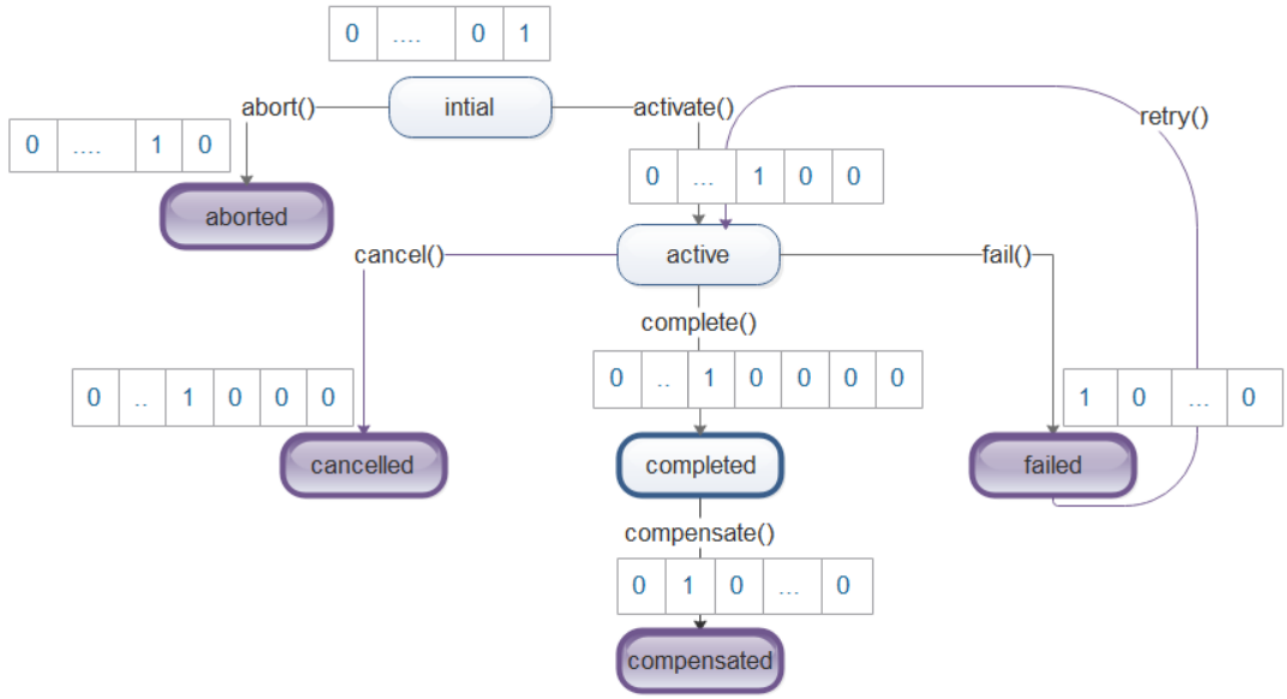


FIGURE 3.5 – Vecteur d'état

approche nous utilisons ces vecteurs dans la template EJS, il est nommé par **StateVector_ActivityName** et par défaut il est inutilisé à "0". Lorsque le jeton se met sur le point entrant de la première activité, le modèle BPMN-to-Solidity génère la première fonction pour commencer l'exécution de processus et mettre à jour le vecteur d'état vers l'état initial. Puis a chaque transition d'une activité vers l'autre ce vecteur sera changé selon l'état.

3.4.3.2 Définitions des fonctions des transitions

Nous implémentons les transitions **abort(), cancel(), complete(), fail() et retry()** qui permettent à l'activité de se déplacer d'un état à un autre en tant que fonctions dans le «Process S.C». Ces fonctions mettent à jour la valeur des vecteurs d'état des activités. Chaque fonction de transition prend en paramètre la valeur de vecteur d'état de l'activité en courant.

abort_task() : permet d'abandonner l'exécution d'une activité avant d'être activé et le fait passer ainsi de l'état initial à l'état abandonné.
cancel_task() : permet d'annuler une activité en cours de son exécution et le fait ainsi transité de l'état activé vers l'état annulé.
complete_task() : permet de marquer la terminaison d'une activité avec succès et de le faire ainsi transiter de l'état activé vers l'état terminé.
fail_task() : permet de marquer l'échec d'une activité et le fait ainsi transité de l'état activé vers l'état échoué.
compensate_task() : permet de compenser sémantiquement le travail effectué par une activité et le fait ainsi transité de l'état terminé vers l'état compensé.
retry_task() : permet d'activer une activité après son échec et le fait ainsi transité de l'état échoué vers l'état activé.

3.4.3.3 Définition des fonctions d'état de terminaison transactionnelles

Selon la propriété transactionnelle d'une activité nous définissons la fonction nécessaire.

— **Cas d'échec d'une propriété pivot :**

Une fois que l'activité de modèle BPMN qui apporte une propriété transactionnelle **Pivot** est échoué au cours de son exécution, une fonction qui est nommé **ActivityName_fail** sera généré avec l'indice d'activité dans le processus comme illustré dans l'algorithme 1 cette fonction permet de récupérer d'abord la valeur d'emplacement de jeton et la valeur de vecteur d'état de l'activité courant depuis le processus, et faire appel à la fonction de transition correspondante qui nous appelons **fail_task**, cette fonction à son tour exécute l'état le processus pour passer de l'état actuel (état **activé**) vers l'état d'échec (état **fail**). Puis elle fait appel à la fonction **Step** pour générer le contrôle de flux d'activité.
Quelque soit l'état de l'activité est utilisé pour capturer l'état de processus dans la blockchain. l'exécution de la fonction externe **ActivityName_fail** nécessite une interaction avec un oracle, lorsque l'activité est dans l'état échoué le smart contract oracle (Worklist) est invoquée et effectue l'appel de réponse correspondante et la tâche est marquée comme **failed**.

Algorithm 1: Cas d'échec d'une activité pivot

```
fonction ActivityName_fail (elementIndex : uint)
  var (tmpMarking,tmpSateVector) = (marking,StateVector_ActivityName);
if elementIndex == uint nodeRealIndex(nodeId) then
    require(msg.sender == worklist && tmpStateVector & uint(4) != 0) ;
    StateVector_ActivityName = fail_task(tmpStateVector) ;);
    step(tmpMarking | uint postMarking(nodeId),StateVector_ActivityName);
end
```

— **Cas d'échec d'une propriété compensable :**

Dans le cas qu'une activité de modèle BPMN apporte une propriété transactionnelle **Compensatable** est échouée au cours de son exécution, une fonction qui est nommé **ActivityNamecp_complete** se génère cette fonction permet de récupérer d'abord la valeur d'emplacement de jeton et la valeur de vecteur d'état de l'activité courant depuis le processus et faire appel à la fonction de transition correspondante qui nous appelons **compsate_task** de l'algorithme 2, cette fonction à son tour exécute l'état la processus pour passer de l'état actuel (état **terminé**) vers l'état d'échec (état **compensé**). Puis elle fait appel au fonction **Step** pour générer le contrôle de flux d'activité.

L'exécution de la fonction externe **ActivityNamecp_complete** nécessite une interaction avec un oracle, lorsque l'activité est dans l'état échoué le smart contract oracle (Worklist) est invoquée et effectue l'appel de réponse correspondante et la tâche est marquée comme **compensated**.

Algorithm 2: Transition vers l'état compensé

```
Result: le vecteur d'état
fonction Compensate_task (elementIndex : uint) :uint
 if tmpStateVector & uint(16) != 0 then
    l'activité n'est plus à l'état terminé;
    tmpStateVector &= uint(~16);
    l'activité a l'état compensate;
    tmpStateVector |= uint(32);
    Retourner la vecteur d'état avec la nouvelle valeur;
    return (tmpStateVector);
end
```

— **Cas d'une propriété rejouable :**

Dans le cas d'échec d'une activité rejouable une fonction **ActivityName_fail** se génère. Cette fonction à son tour appelle la fonction de transition **retry_task()** pour passer de l'état actuel (état **fail**) vers l'état **activé**. Puis elle faire appelle au fonction **Step** pour générer le contrôle de flux d'activité.

— **Cas d'une propriété compensable+rejouable :**

Dans le cas d'échec d'une activité compensable + rejouable l'activité peut être compensé après sa terminaison donc elle est dans le même cas avec une activité compensable (faire appel à la fonction ActivityNamecp_complete) plus qu'elle peut réactivé après un échec donc elle est dans le même cas avec une activité réjouable (faire appel à la fonction Activity_fail).

3.5 Gestion d'échec des exécutions des activités

L'état actuel d'une tel activité de processus est capturé via le variable StateVector_ActivityName.

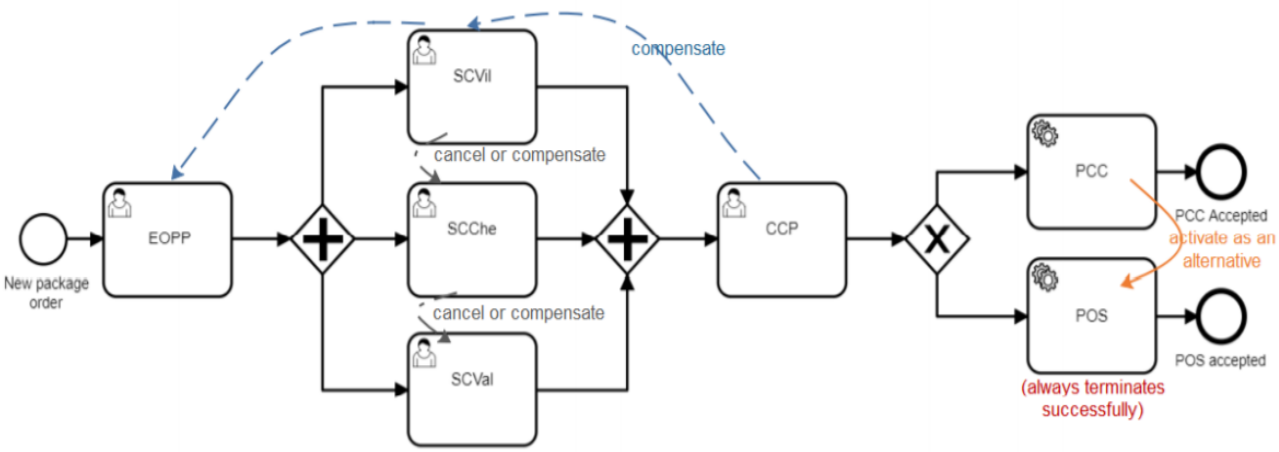


FIGURE 3.6 – Flux de contrôle d'un exemple d'un modèle BPMN

3.5.1 Cas d'échec d'une activité avec la propriété pivot

Lorsqu'une activité de processus échoue, une fonction d'échec sera générée, cette fonction permet de mettre à jour le vecteur d'état de la tâche de activé vers échoué et de réinitialiser le jeton et le mettre sur le bord entrant de l'activité en cours puis elle fait appel la fonction de contrôle de flux qui permet d'exécuter ces actions :

- Faire une compensation pour toutes les activités compensables précédentes jusqu'à arriver au pivot précédent.
- S'il y a une alternative après le pivot précédent on passe a l'alternative suivante sinon on mettre les activités suivantes a l'état abandonné.

Selon l'exemple de la figure 3.6 en cas d'échec d'activité CCP, on componse les activités SCVIL et EOPP. (car ils ont la propriété compensatable) et on modifie l'état des activités suivantes (PCC et POS) vers l'état abandonné.

3.5.2 Cas d'échec d'une activité avec la propriété compensable

Dans le cas d'une activité compensable échoue une fonction de **cp_complete()** est généré. Au cours de son exécution une fonction Step est appelé pour exécuter ces actions :

- Si l'activité compensable est située dans une branche parallèle on compense les activités terminées avec la fonction **compensate_task** qui sera généré pour modifier l'état de l'activité. Puis on génère une fonction d'annulation **cancel_task** pour les activités qui sont en cours d'exécution.
- Sinon on compense les activités précédentes jusqu'à pivot.

Selon l'exemple de modèle de la figure 3.6 en cas d'échec de l'activité SCVil, on annule l'exécution des activités en parallèle ou les compensent si elles ont déjà terminé leur travail.

3.5.3 Cas d'échec d'une activité avec la propriété rejouable

Dans le cas d'échec d'une activité avec la propriété rejouable on le réactive jusqu'à se termine avec succès. Dans ce cas nous attribuons un seuil de réactivation de l'activité. A chaque fois une fonction start() est appelée par le smart contract oracle (worklist) lorsqu'un utilisateur exécute la tâche. et une fonction de transition activate_task sera lancé pour passé de l'état fail à l'état activé.
Selon l'exemple de modèle de la figure 3.6 en cas d'échec de tâche PCC, il faut activer le tâche POS comme une alternative.

3.5.4 Cas d'échec d'une activité avec la propriété compensable et rejouable

Les activités avec les propriétés **compensatable+retriable** sont principalement des activités **compensatable** avec une transition **retry()** qui faire une transition a l'activité de l'état **fail** vers l'état **active**. Pour gérer l'échec dans ce cas, si l'activité échoue : en tant qu'on n'a pas atteint le threshold, on le réactive et si on atteint le threshold, elle passe à l'état fail.

Partie2 : Extension pour le cas d'exécution de TBP collaboratif sur une blockchain privée avec permission

Des nombreux chercheurs ont travaillé sur le processus métier basé sur la technologie de la blockchain publique sans permission, Tout ces travaux admettent l'applicabilité de l'utilisation du blockchain pour faciliter la gestion des processus métiers collaboratifs mais ils souffrent des défis qui nous pouvons rencontrer lors de développement (1) Une fois qu'un smart contract est déployé, personne ne peut modifier le contenu. Les développeurs ne peuvent pas corriger les bugs ou les erreurs d'un contrat existant, ce qui peut faire le bonheur aux pirates informatiques. Les smart contracts doivent donc être soigneusement étudiés et audités avant leur déploiement, (2) Une faible performance en termes de capacité à traiter un débit élevé des demandes des transactions, (3) Les données sont accessibles au public de telle sorte qu'ils sont entièrement reproduits parmi tous les pairs.

De plus les systèmes de blockchain publiques sans permission ont mentionné certains conflits importants tels que **le contrôle de confidentialité (Privacy) et mise à l'échelle (Scalability)**. Pour cette raison on ne peut pas considérer la blockchain publique sans permission comme une base appropriée pour la BPM puisqu'elle ne peut pas assurer la confidentialité des identités des participants, de leurs actifs, leurs smart contracts ainsi que leurs transactions. Nous avons besoin des certaines procédures d'accès pour nos smart contracts déployés, et nous devrions être en mesure pour déterminer qui peut accéder aux ressources des transactions ou des smart contracts spécifique dans le réseau blockchain.

Et puisque nous avons dans le cadre de BP collaboratif plus précisément dans la chorégraphie de BP, les BC privées avec permission sont plus adaptés pour l'exécution de notre processus puisqu'elles permettent une meilleure **confidentialité** et **mise à**

l'échelle.

En résumé, pour la plupart des cas d'utilisation des processus métiers inter-organisationnels, les organisations sont susceptibles de préférer les solutions avec permission, où la participation est soumise à l'identification de toutes les parties, à la confidentialité des données sensibles et au respect des règles du système. Ainsi que la persistance, la validité et l'auditabilité dans une blockchain avec permission peuvent être obtenues à un certain niveau en fonction des organisations impliquées dans le réseau. D'un autre côté, l'exécution des processus métiers transactionnels collaboratifs sur une blockchain avec permission devient un sérieux défi d'intégration pour les entreprises qui doivent améliorer leurs performances pour avoir une exécution correcte des processus métiers dans un environnement qui donne l'avantage de confidentialité des informations et la sécurité de processus.

Notre proposition se base sur le travail de Falazi et al. et l'étendre. Les auteurs dans [GF19] permettent l'implémentation d'un processus métier sous forme d'un smart contract en blockchain privée avec permission. Nous adoptons cette approche pour exécuter une collaboration de processus en ajoutant de plus l'aspect transactionnel à fin de gérer les échecs introduits dans les activités de processus métier.
Nous commençons par la description un aperçu sur l'architecture de Falazi, puis nous décrivons notre proposition pour exécuter un TBP collaboratif sur une blockchain privée avec permission.

3.6 Exécution un processus métier transactionnel collaboratif sur une blockchain privée avec permission

Dans cette partie nous avons proposé de développer notre approche dans le cas d'exécution de processus métier collaboratif dans une blockchain privée avec permission. Nous avons étendu le système de Falazi [GF19, GFL20] en ajoutant l'aspect transactionnel. Dans cette approche, un modèle de processus développé en un langage BPMN, permet à une entreprise de décrire visuellement quand et dans quelles conditions elle invoque des fonctions de smart contract résidant dans un système blockchain auquel elle est connecté, il modélise également les règles métier qui régissent ces appels. Un tel modèle de processus est ensuite instancié et exécuté sur un système de gestion de processus métier (BPMS) qui traduirait les constructions modélisées en messages réels et en appels d'API. Ce système se compose en trois parties :

− La Blockchain Access Layer **(BAL)** est un composant logiciel extensible qui unifie l'accès à divers systèmes de blockchain publics et privées qui prend en charge l'exécution de modèles de processus métier qui doivent communiquer avec ces blockchains.
− Un protocole de Smart Contract Invocation Protocol **(SCIP)**, présente l'interaction avec les smart contracts des différentes blockchains. C'est un schéma URI générique pour identifier les fonctions de smart contract à la fois pour la blockchain avec et sans permission.
Nous nous intéressons dans notre travail à la partie de SCIP où nous intégrons notre solution.

3.6.1 Aperçu sur SCIP

Les fonctions de smart contract sont adressables en spécifiant un ou plusieurs niveaux hiérarchiques, puis la fonction elle-même. Par conséquent, ces fonctions sont traités comme des ressources utilisent la norme Web Unified Resource Identifier (URI) pour les identifier. Le Smart Contract Invocation Protocol (SCIP) présente un schéma d'identifiant de ressource qui permet d'interagir avec les smart contracts de différentes blockchains. Ce protocole prend en charge l'appel des fonctions de smart contract, surveillance des exécutions de fonctions, des événements émis et de la qualité des transactions, ainsi que l'interrogation d'une blockchain. Il est accompagné d'une implémentation prototypique d'un endpoint SCIP sous la forme d'une passerelle.

Spécification de SCIP

Le protocole définit un total de quatre méthodes prises en charge différentes qui peuvent être appliquées aux smart contracts :

L'invocation des fonctions de smart contract : La demande d'appel est effectuée via la méthode Invoke, qui permet à une application externe d'appeler une fonction de smart contract spécifique.

L'abonnement aux appels de fonctions ou à la surveillance en direct des occurrences d'événements : Cette requête est exécutée à l'aide de la méthode Subscribe, en particulier cette requête facilite la surveillance en direct des smart contracts en permettant à une application cliente de recevoir des notifications asynchrones sur les occurrences d'événements ou les appels des fonctions de smart contract.

La désinscription des appels des fonctions ou des événements de surveillance en direct : Cette demande est modélisée à l'aide de la méthode Unsubscribe, et utilisée pour annuler explicitement les abonnements d'un client, précédemment générés à l'aide d'appels de la méthode Subscribe.

L'interrogation des invocations des fonctions passés ou des occurrences d'événement : Cette requête est effectuée via la méthode Query, son but est de permettre à l'application cliente d'interroger les occurrences précédentes d'appels d'événement ou de fonction.

3.6.2 Aperçu sur de BAL

Sous le point de terminaison SCIP, le noyau BAL (Blockchain Access Layer) fournit la logique de traitement des demandes et d'envoi de rappels (callback). Une unité d'analyse et d'évaluation d'expression booléenne prend en charge le champ de filtre de certaines méthodes SCIP.

Security Manager : permet d'authentifier les applications clientes et gérer les signatures des messages de demande Invoke.

Blockchain Manager : permet de coordonner le travail des autres composants et communique avec la couche d'adaptateur en dessous. Pour chaque blockchain prise en charge (Ethereum ou Hyperledger Fabric), il été implémenté un module adaptateur qui gère les spécifications du système blockchain correspondant. Ce module gère la façon dont les smart contracts sont invoqués, la façon dont les paramètres sont encodés / décodés, la façon dont les événements peuvent être surveillés et interrogés, etc.
Les adaptateurs communiquent directement avec un noeud blockchain, et par conséquent, incluent un client de protocole spécifique à la blockchain pour effectuer des actions spécifiques à la blockchain.

Expression Parser : prend en charge le champ de filtrage de certaines méthodes SCIP.

Callback Manager : est chargé de renvoyer le résultat de chaque opération demandée au point final spécifié par l'application externe.

Le Subscription Manager : est responsable de la corrélation des messages de demande reçus des applications externes avec leurs réponses.

3.6.3 Solution proposée

Notre solution vise à assurer la confiance et garantir la coté de confidentialité dans l'exécution des TBP collaboratifs. Pour cela nous avons étendu le système de Falazi pour qu'on puisse avoir une exécution correcte des processus métiers transactionnels collaboratifs sur une blockchain privée avec permission c'est à dire que nous avons pris en charge les cas d'échecs des tâches.
Sur cette base nous avons proposé d'améliorer ce système avec les mécanismes transactionnels pour atteindre notre objectif.

Dans une première étape nous nous essayons de modifier notre processus métier entrant, le modèle BPPMN d'exécution d'entrée n'est plus une orchestration mais une collaboration de processus métier où il existe plusieurs processus qui interagissent entre eux et chaque processus implémente la logique métier d'un participant.
Notre cas d'utilisation dans cette partie est une interaction entre un touriste est un château pour la vente et achat d'une billet électronique.

Puis nous avons enrichi ce modèle de processus métier avec l'aspect transactionnel c'est-à-dire qui nous avons ajouté les propriétés transactionnelles nécessaires (Pivot, compensable ou rejouable). Ensuite nous avons transformé ce processus en smart contract et l'exécuter dans la blockchain privée HyperledgerFabric.

Notre proposition vise à proposer une contribution pour garantir une exécution fiable des collaborations des processus métiers transactionnels en une blockchain privée avec permission.

La figure suivante 3.7 présente le modèle de processus métier collaboratif qui nous avons considéré pour développer notre solution.

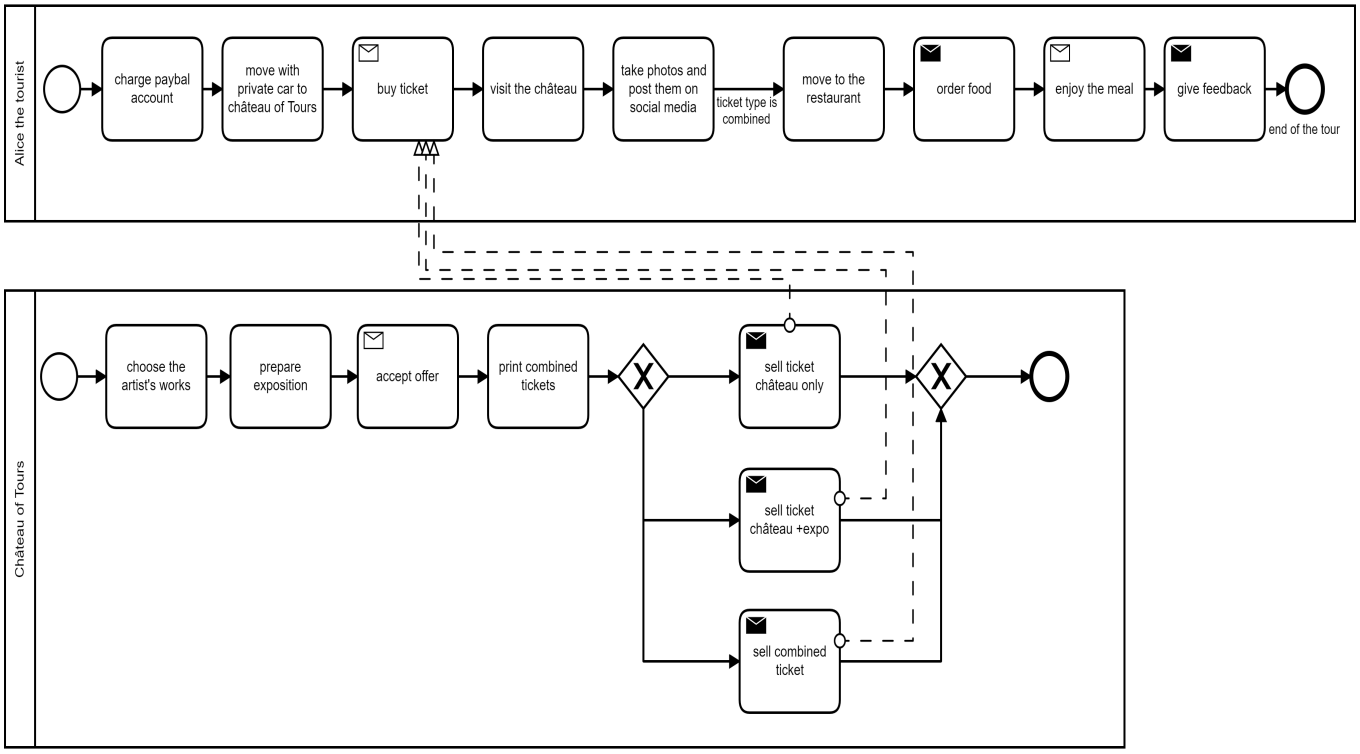


FIGURE 3.7 – Modèle de processus métier collaboratif

A cet égard nous avons développé un smart contract sous le nom de **InteractionSC** qui implémente les fonctions métier de ce modèle. Une fonction pour émettre le ticket qui va être invoqué de la part du château et une fonction pour acheter ce ticket de la part du touriste.
La fonction sous le nom de **sellCombinedTicket** permet de créer l'instance de ce ticket elle émit le ticket de la part du château et l'ajouter à la liste dans l'état du grand livre, et la fonction du nom **BuyCombinedTicket** permet de récupérer le ticket actuel à l'aide des champs clés fournis, valider le propriétaire puis de mettre à jour la liste des billets.

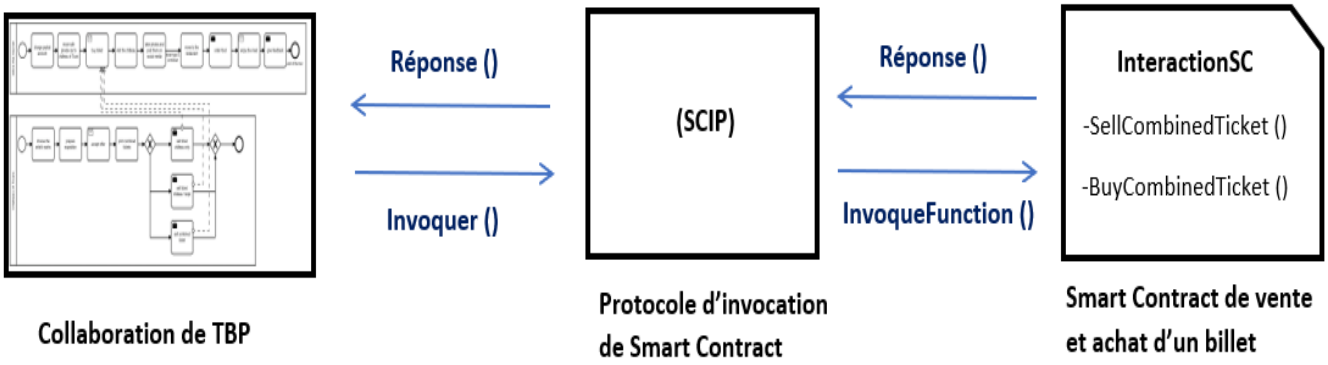


FIGURE 3.8 – Communication de TBP avec SC à travers SCIP

Pour lier le processus métier avec notre SC on va utiliser le protocole SCIP de [GFL20]. La figure 3.8 présente un schéma qui montre une interaction courante entre le modèle de processus métier et le smart contract développé à travers le protocole SCIP impliquant l'appel d'une fonction de SC. Un appel d'une fonction est envoyé de la part de TBP, puis à l'aide de SCIP une fonction d'invocation est déclenché vers le smart contract, et après l'exécution de cette fonction le SC envoie une réponse.

3.7 Conclusion

Dans ce chapitre, nous avons présenté notre approche d'exécution des processus métiers transactionnels en smart contract, en particulier nous avons décrit l'implémentation de comportement transactionnel dans une blockchain publique sans permission dans une première partie puis une étude théorique pour la continuité de notre approche qui présente l'exécution de TBP dans une blockchain privée avec permission. Dans le prochain chapitre, nous présenterons la mise en oeuvre et les scénarios implémentés dans les deux cas.

Chapitre 4

Mise en oeuvre

4.1 Introduction

Dans ce chapitre, nous passons à l'étude pratique de notre solution proposée qui présente une exécution des processus métiers transactionnels sur la blockchain. En premier, nous présentons les plateformes d'exécution qui nous avons étendues pour y intégrer notre TBP. Par la suite, nous présentons les scénarios implémentés avec les traces d'exécutions. Pour ce faire, nous avons proposé deux solutions : la première consiste à exécuter un processus métier transactionnel d'orchestration sur une blockchain publique sans permission et la deuxième consiste à exécuter un processus métier transactionnel collaboratif sur une blockchain privée avec permission.

4.2 Platefomes d'éxecution et langages des programmations

L'intégration de processus métier transactionnel dans le smart contract de la blockchain est un sujet qui nous a mené à utiliser les plateformes de blockchain. Dans la suite nous présentons les plateformes utilisées pour développer notre proposition.

4.2.1 Cas de blockchain publique sans permission

Dans ce cas nous avons étendu le système de Caterpillar où nous exécutons un processus métier transactionnel d'orchestration. La spécificité de Caterpillar est que l'état de chaque instance de processus est maintenu sur la blockchain (Ethereum) et que le routage du processus métier est effectué par les smart contracts générés par un compilateur BPMN-to-Solidity.

Le runtime off-chain de Caterpillar et l'interface utilisateur Web sont implémentés dans Node.js et s'appuient sur le compilateur Solidity standard solc-js pour compiler les smarts contracts. Les interactions avec les instances en cours d'exécution de ces smarts contracts sont prises en charge via le client Ethereum geth et la fonctionnalité du runtime off-chain de Caterpillar est exposée via une API REST.

4.2.1.1 Ethereum

Ethereum [1] [Woo14] est une plateforme ouverte et utilisée pour créer des applications décentralisées qui fonctionnent sur la technologie blockchain. Elle a été conçu pour être flexible afin de faciliter la création des nouvelles applications. Ethereum est considéré comme une machine à états basée sur les transactions qui commence à partir d'un bloc de genèse et une chaîne des blocs formée par des transactions entre les états de compte où les transactions sont enregistrées en blocs.

– **Ethereum Virtual Machine (EVM) :** C'est une machine informatique qui fournit un environnement pour l'exécution du code EVM (Code EVM est l'octet-code que l'EVM peut exécuter).
Les smart contracts sont exécutés sur la machine virtuelle Ethereum (EVM), qui est regroupée dans chaque noeud pair, pour chaque contrat, l'EVM alloue une mémoire persistante, appelée stockage, qui est organisé comme une mémoire clé-valeur qui mappe des mots de 256 bits à des mots de 256 bits.
L'EVM a une limite sur le nombre des étapes de calcul qui est défini comme gaz.

– **Noeud :** Un noeud exécute l'implémentation EVM. Les noeuds envoient des transactions à d'autres noeuds et peuvent également exécuter un mineur.

– **Mineur :** Un mineur est un processus qui sélectionne les transactions envoyées par les noeuds et génère le bloc. Les mineurs fonctionnent sur un noeud Ethereum.

– **Exploitation minière :** l'exploitation minière est le processus d'ajout de blocs par preuve de travail qui implique la résolution des problèmes mathématiques complexes.

– **Ether :** Ether est la devise utilisée dans la plateforme Ethereum pour simplifier le calcul au sein du réseau.

– **Go-Ethereum (geth) :** Go-Ethereum [2] c'est une interface de ligne de commande pour exécuter un noeud Ethereum complet implémenté dans le langage de programmation Go. Les fonctions Geth sont accessibles via HTTP en utilisant le protocole JSON-RPC qui est un protocole d'appel de procédure à distance. En installant et en exécutant Geth, nous pouvons participer au réseau direct de la frontière d'Ethereum, à mon vrai éther, au transfert des fonds entre les adresses à la création des contrats et l'envoient des transactions et l'exploration de l'historique des blocs.

– **Adresse :** Un identifiant de 160 bits. Une fois déployé (ou instancié) un smart contract. il est lié à une adresse de hachage unique qui peut être utilisée par des applications externes pour appeler les fonctions publiques du SC.

– **État du compte :** L'état de compte se compose d'un nonce, d'un solde, d'un storageRoot et d'un codeHash. Un nonce représente le nombre de transactions envoyées depuis le compte, le solde est le montant d'Ether détenu par le compte, storageRoot est un hachage de 256 bits du noeud racine d'un arbre de hachage qui encode le contenu de

1. `https://buildmedia.readthedocs.org/media/pdf/ethereum-homestead/latest/ethereum-homestead.pdf` visité le 15-11-2020
2. `https://github.com/ethereum/go-ethereum/wiki/geth`visité le 15-11-2020

stockage du compte et codeHash est le hachage du code EVM du compte. Ce code est exécuté lorsque le compte reçoit un appel de message.

– **Gaz :** Le gaz est la quantité d'ether à payer pour exécuter une transaction particulière dans l'EVM. Son prix est fixé par la transaction et les mineurs peuvent ignorer les transactions dont le prix du gaz est trop bas.
D'un point de vue technique, le smart contract correspond au code qui est déployé sur la blockchain Ethereum. Le coût de déploiement de ce contrat dans Ethereum, qui est proportionnel à sa taille de bytecode, est mesuré dans une unité appelée gaz.

– **Transaction :** Une transaction est une donnée signée par un acteur externe. Les transactions sont enregistrées dans chaque bloc de la blockchain et chaque transaction se compose de nonce, gasPrice, gasLimit, adresse, valeur et données. Nonce est le nombre de transactions envoyées depuis un compte, gasPrice est le montant à couvrir pour tous les frais de calcul encourus à la suite de l'exécution de cette transaction, gasLimit est la quantité maximale de gaz qui doit être utilisée pour exécuter cette transaction, à l'adresse du compte contractuel du destinataire, la valeur est la quantité d'éther transféré aux comptes des destinataires et les données se composent soit du code de contrat pour déployer un nouveau contrat, soit des paramètres pour activer une fonction d'un contrat existant.

– **Genesis Block :** C'est le premier bloc de la blockchain qui définit la configuration initiale, la difficulté d'extraction et la limite de gaz par bloc.

– **Ganache :** Ganache [3] est une blockchain personnelle pour le développement rapide d'applications distribuées Ethereum. Nous pouvons l'utiliser pendant tout le cycle de développement ; vous permettant de développer, déployer et tester vos dApps dans un environnement sûr et déterministe.
Ganache UI est une application de bureau prenant en charge les technologies Ethereum. Une version Ethereum de ganache est disponible en tant qu'outil de ligne de commande : ganache-cli (anciennement TestRPC).

4.2.1.2 Solidity

Solidity [4] est un langage fortement typé, et sa syntaxe ressemble à JavaScript. Un contrat dans Solidity est défini de la même manière que les classes dans les langages orientés objet de type Java. Ainsi, la définition des contrats comprend généralement des propriétés persistantes (c'est-à-dire l'état du contrat) et des fonctions pour les interroger et les manipuler et conçue pour cibler la machine virtuelle Ethereum.

– **Compilateur de solidité solc-js :** solc-js [5]présente la liaison JavaScript pour le compilateur Solidity. Il existe deux façons d'utiliser solc soit via une API de haut niveau donnant une interface uniforme à toutes les versions du compilateur ou via une API de bas niveau donnant accès à toutes les interfaces du compilateur, qui dépendent de la version du compilateur.

3. `https://www.trufflesuite.com/docs/ganache/overview`
4. `https://docs.soliditylang.org/en/v0.7.2/contracts.html` visité le 15-11-2020
5. `https://github.com/ethereum/solc-js`visité le 15-11-2020

4.2.2 Cas de blockchain privée avec permission

Pour exécuter un processus métier transactionnel dans une blckchain privée avec permission. Nous avons choisi la plateforme de blockchain Hyperledger Fabric.

4.2.2.1 Hyperledger Fabric

Hyperledger Fabric [EA18] est l'une des plateformes DLT développés au sein de la communauté des hyperlires. Le Fabric est également un registre immuable, maintenu au sein d'un réseau décentralisé de noeuds. C'est un réseau avec permission, la confidentialité des transactions et la gestion des identités sont les caractéristiques les plus importantes du Fabric.

– **Pair(Peer) :** La blockchain est un réseau distribué de noeuds (dans ce cas des pairs) qui fait des pairs des blocs de construction d'un réseau blockchain. Dans le fabric, un pair par définition peut héberger plusieurs instances de registres et de smart contract. Les pairs sont classés en trois catégories qui sont données ci-dessous :

- **Endorsing Peers :** Sont crées pour recevoir les propositions de transaction de l'application client pour les approbations. Des smarts contracts doivent être présents pour exécuter la proposition.
- **Committing Peers :** Sont crées pour valider les transactions et maintient l'état du grand livre.
- **Ordering peers :** C'est sont des Types spéciaux de pairs où les transactions approuvées sont reçues, classées en blocs et distribuées aux pairs de validation pour la validation et la mise à jour de leurs registres respectifs.

– **Ledgers :** Le terme Ledgers dans l'hyperledger fabric est associé à deux parts une base de données d'état et un journal des transactions. La base de données d'état enregistre les valeurs actuelles des actifs et permet au développeur d'accéder facilement aux valeurs au lieu de parcourir toute la chaîne. Fabric permet la prise en charge de deux technologies DB LevelDB et CouchDB. LevelDB est la valeur par défaut et traite les objets simples et ce dernier traite des documents JSON, ce qui le rend adapté aux objets complexes et permet des requêtes riches. Le journal des transactions n'est rien d'autre qu'une blockchain qui enregistre toutes les transactions qui aboutissent aux valeurs actuelles des actifs.

– **Membership service provider(MSP) :** Dans chaque blockchain avec permission, toutes les interactions entre les noeuds doivent être authentifiées, c'est-à-dire que chaque noeud du réseau doit avoir une identité afin que la vérification d'authentification puisse être effectuée.
Le MSP dans la structure est le composant qui fournit les informations d'identification ou certifie à tous les clients et pairs participants au réseau et conserve leur identité. Pour la fourniture de certificats, le MSP utilise une autorité de certification (CA) pour délivrer et vérifier les certificats. Par défaut, Fabric-CA est utilisé dans la blockchain Fabric.

– **Organisations :** Aussi appelées membres, les organisations sont un groupe de pairs. Chaque organisation est mappée à son propre MSP qui donne l'identité à tous les pairs appartenant à sa propre organisation. Un ensemble d'organisations en réseau forment un consortium.

– **Channel :** Les canaux permettent un ensemble spécifique de pairs d'une ou plusieurs organisations et des applications pour communiquer entre elles au sein d'un réseau. Chaque pair sur le channel est associé à un registre unique. Dans un réseau Fabric, il peut y avoir plusieurs canaux, ce qui rend les canaux quelque peu similaires aux blockchains privées.

– **Service de commande (ordering) :** Composé d'un groupe de noeuds de commande avec la tâche de diffuser les transactions approuvées aux pairs de validation et d'assurer l'ordre de toutes les transactions dans un bloc de livraison. À l'heure actuelle, le tissu prend en charge trois types de service de commande, à savoir SOLO [SB18], KAFKA [SB18] et RAFT [DO14] .

– **Transaction :** La gestion des transactions en hyperledger est presenté en trois étape [Abh19]

- **Phase d'execution** L'application client envoie une proposition de transaction à un ou plusieurs endosseurs. Les pairs endosseurs exécutent la transaction sur la base de données d'état local de pair, puis approuvent la transaction et la renvoient au client. Le client collecte les approbations (endorsement) des pairs et les soumet au service de commande (ordering service).
- **Phase de commande (ordering)** Lors de cette phase, les noeuds orderer valident les transactions par rapport à leurs politiques d'approbation (endorsement policy) respective. La prochaine étape consiste à s'entendre l'ordre des transactions selon les mécanismes de SOLO, Kafka et RAFT. Les orderers regroupent ensuite plusieurs transactions en blocs et ces blocs sont diffusés aux pairs de validation.
- **phase de validation** Chaque bloc entré dans la phase de validation exécutera ces étapes (1) la validation des transactions par rapport à leurs politiques d'approbation respective est effectuée. (2) Toutes les transactions d'un bloc subissent un contrôle du conflit lecture-écriture où l'ensemble de lecture est comparé aux valeurs respectives de l'ensemble d'écritures. Si une transaction enfreint les règles de conflit, elle est marquée comme invalide mais elle reste dans le bloc avec une balise invalide. (3) Le DB d'état local de paires est mis à jour dans l'ensemble d'écritures de la transaction et le bloc est ajouté au journal des transactions de pair.

– **Docker :** les principaux composants de Hyperledger Fabric sont les images docker [AB18]. Fabric exécute chaque chaincode dans un conteneur Docker distinct. Docker assure l'isolation entre différents chaincode s'exécutant sur la même machine.

4.2.2.2 JavaScript

Nous avons choisir le javascript pour developper notre chaincode. C'est un langage de script léger, orienté objet. Le Hyperledger Fabric prend également en charge Javascript pour exécuter des smart contract et Hyperledger Fabric Client SDK pour Node.js est utilisé pour interagir avec le réseau blockchain.

4.3 Les scénarios implémentés

Au cours de notre travail nous avons implémenté deux scénarios un scénario pour un touriste qui veut commander un paquet personnalisé suite à sa visite aux châteaux de Tours, ce scénario est représenté sous forme d'un processus métier transactionnel d'orchestration et il a été développé en Blockchain Ethereum. Tandis que le deuxième scénario présente une interaction entre un touriste qui va acheter un billet combiné et le château de tours qui va vendre ce billet, scénario est représenté sous forme d'un processus métier transactionnel collaboratif et il a été développé en blockchain Hyperledger Fabric.

4.3.1 Processus de commande de paquet touristique personnalisé

Le scénario de processus de commande de paquet touristique personnalisé a été exécuté a l'aide du système de Caterpillar qui a été réalisée avec l'utilisation du framework Ethereum pour le réseau blockchain, ExpressJS pour le serveur backend REST et Angular et Typescript pour l'application client Web. La figure 4.1 illustre l'exemple ce scénario.

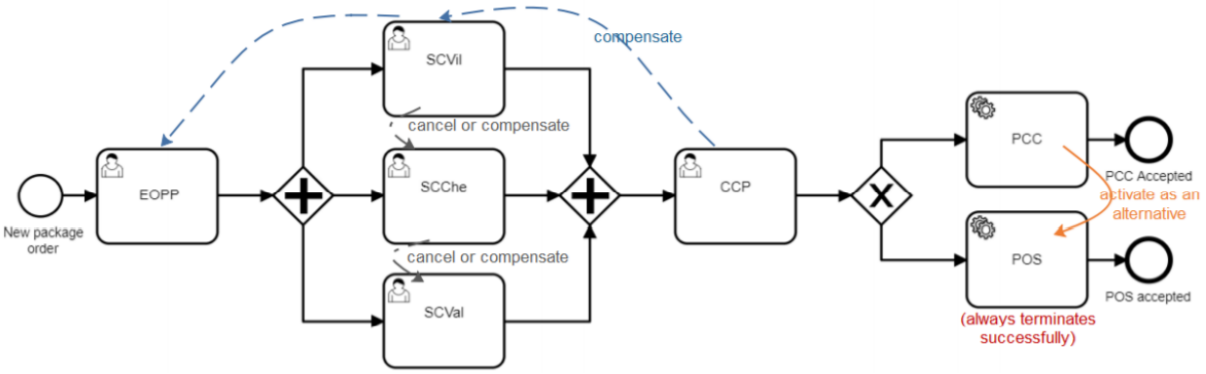


FIGURE 4.1 – Processus de commande d'un packet touristique personnalisé

Ce processus présente une modélisation de scénario d'un touriste qui souhaite visiter les châteaux «along the river» et conçoit et achète ainsi son forfait personnalisé. Il sélectionne les trois châteaux qu'il souhaite inclure dans le forfait comme le château de Villandry, le château de Chenonceau. Puis, après avoir confirmé son choix de forfait, il paie en utilisant l'un des modes de paiement disponibles : carte bancaire ou paiement sur place.

Nous avons essayé d'exécuter ce processus dans l'outil Caterpillar en prenant en compte le cas d'échec de ces activités. En premier étape pour chaque activité de ce processus nous ajoutons la propriété transactionnelle nécessaire pivot,compensatable,retriable ou compensatable+retriable) comme illustré dans la figure suivante 4.2. Grâce à ces propriétés nous pouvons savoir l'état de chaque activité en cours d'exécution.
Puis notre approche met en oeuvre le TBP présenté ci-dessous où son flux de contrôle son flux transactionnel sont donnés en entrée.
La figure suivante présente l'interface client de ce système qui contient les outils de modélisation / visualisation fournie par Camunda où nous importons notre processus pour l'exécuter.

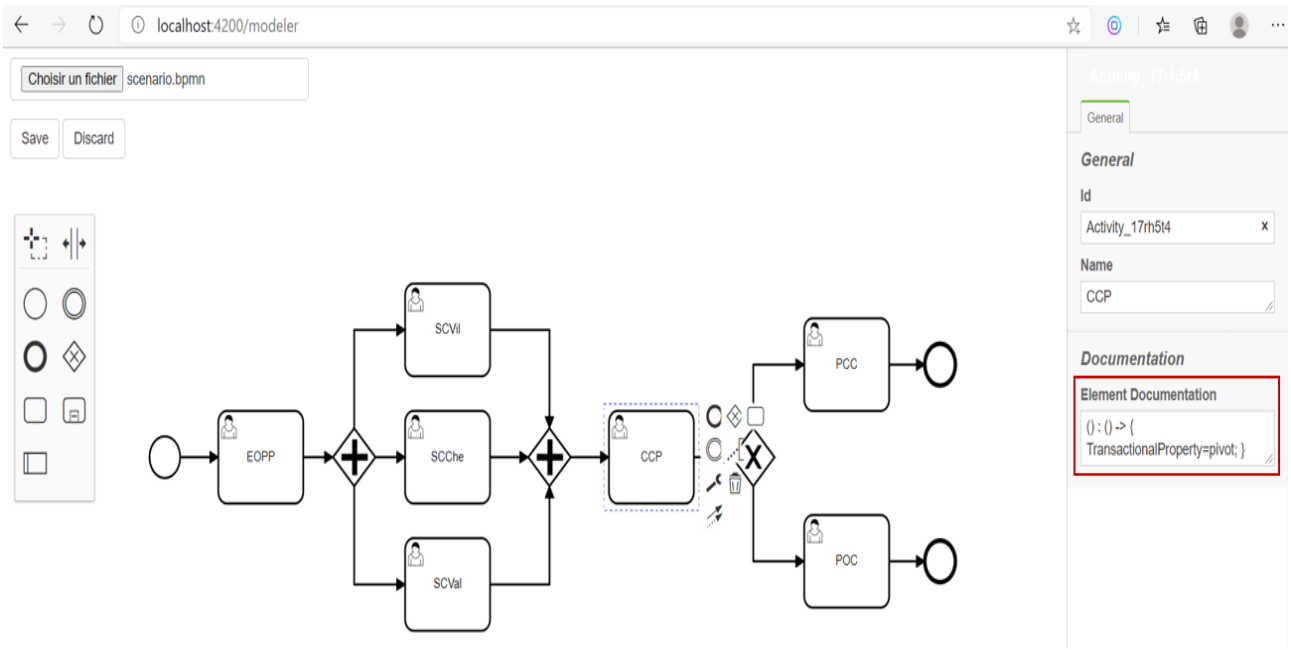


FIGURE 4.2 – Exécution de Processus métier transactionnel sur Caterpillar

En cliquant sur save le modèle passe au système de Caterpillar, il est compilé en un ensemble des smart contracts Solidity avec une structure des données qui comprend des informations (nom de la méthode, index d'élément, type d'élément, propriété de l'élément,etc) permettant de mapper les éléments de modèle BPMN au code généré. Ensuite, l'ensemble de ces smart contracts générés se déployer sur Ethereum comme illustré dans la figure 4.3

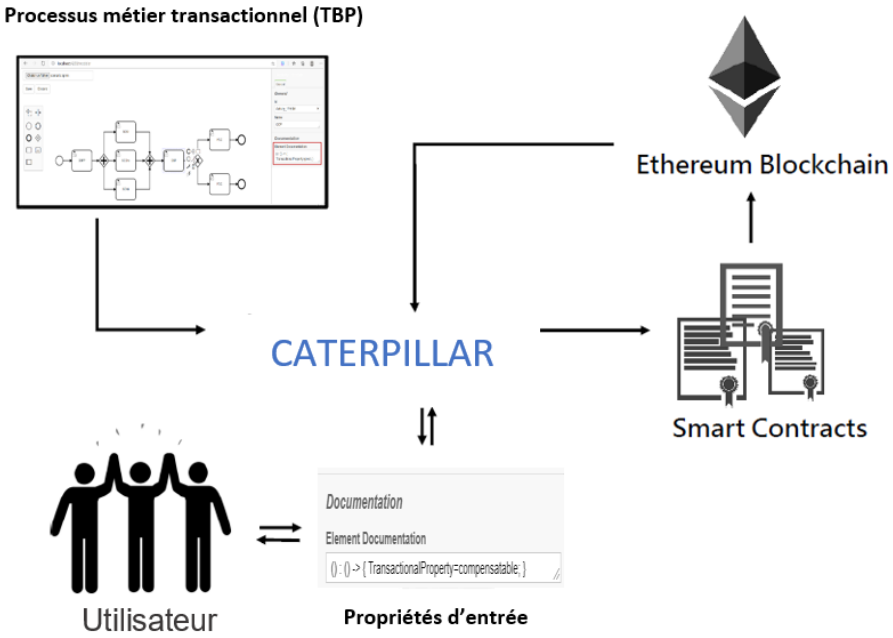


FIGURE 4.3 – Concept d'exécution

Par défaut, Caterpillar a été configuré pour tester l'exécution sur Ganache CLI. Par ailleurs, il faut démarrer le serveur CLI Ganache et le client MongoDB pour stocker et accéder aux métadonnées produites avant d'exécuter les applications Caterpillar Core.

4.3.2 Vente et achat de billet combiné

Dans le cas d'exécution de modèle BPMN transactionnel en blockchain privée avec permission, le scénario de la figure 4.4 a été réalisée avec l'utilisation du framework Hyperledger Fabric. En effet ce scénario est une traduction d'un processus métier collaboratif où chacun de château de tours et de touriste présente une Pool dans le processus, "Sell Ticket" et Buy Ticket" sont deux activités dans le processus.

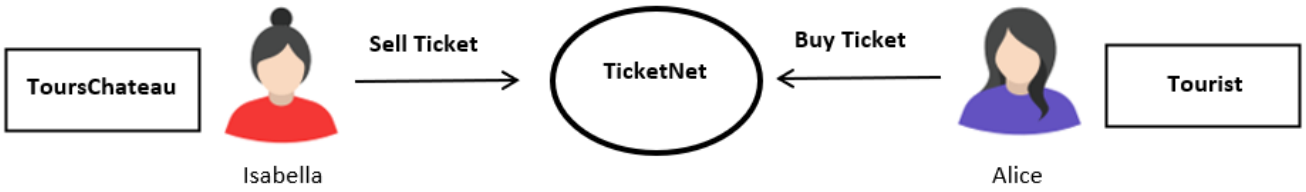


FIGURE 4.4 – Vente et achat de billet combiné

Nous avons créé ce scénario dans Hyperledger Fabric, il implique deux organisations un touriste (Alice) et un administrateur de Château de tours (Isabella) qui utilisent TicketNet, un réseau de ticket combiné basé sur Hyperledger Fabric, pour émettre et acheter du ticket combiné.
Un ticket combiné c'est un ticket qui permet au touriste de visiter plusieurs points d'intérêt avec le même ticket comme par exemple visiter le château de Tours et voir une exposition d'un artiste qui a lieu dans le même château ou un autre château. les fonctions de Buy et sell de ce ticket combiné sont le coeur du smart contract (**SellCombinedTicket() et BuyCombinedTicket()**). Ils sont utilisés par les applications pour soumettre des transactions qui émettent, achètent le ticket combiné dans le grand livre.

Etape1 : Configuration de réseau Hyperledger Fabric

– Génération de matériel cryptographique

Comme expliqué ci-dessus, la structure Hyperledger est un réseau autorisé et chaque composant du réseau à une identité. Le binaire cryptogène fourni par le Fabric est utilisé pour générer tous les certificats fournis par le fichier de configuration qui indique le nombre des organisations, le nombre des pairs associés à chaque organisation, etc. Tous les certificats utilisés sont générés dans un répertoire (crypto-config dans notre cas).

– Configuration et création de channel

Dans le cas de notre projet, un consortium des deux organisations avec un seul channel les reliant est utilisé. L'outil binaire configtxgen est utilisé pour générer les fichiers de configuration pour les canaux et les anchor-peers.
En premier étape nous créons le fichier de configuration (configtx.yaml) pour spécifier les détails des organisations qui nous utilisons dans le réseau. Trois sorties différentes sont générées dans cette section sont enregistrés sous la répertoire channel-artifacts à savoir :

- genesis.block.
- channel.tx.
- Les fichiers de transaction Anchorpeer pour les deux organisations impliquées.

– Création des composants de réseau

Notre réseau blockchain a été mis en place avec deux organisations Org1(Tourist) et Org2(ToursChateau) et un pair par organisation. En raison de la grande complexité de la procédure globale pour configurer tous les modules nécessaires pour que la blockchain soit opérationnelle, un script bash est utilisé. Tous les différents composants du réseau doivent être déployés dans des conteneurs docker isolés, c'est-à-dire que chaque noeud pair s'exécute sur un conteneur Docker différent.
Docker-compose.yaml est un fichier de configuration qui est utilisé pour déployer le réseau Hyperleger Fabric dans les conteneurs Docker, il est nécessaire pour faire passer les noeuds réseau (Peer et Orderer) en tant que conteneurs.

```yaml
peer0.org1.example.com:
  container_name: peer0.org1.example.com
  image: hyperledger/fabric-peer:latest
  environment:
    - CORE_VM_ENDPOINT=unix:///host/var/run/docker.sock
    # the following setting starts chaincode containers on the same
    # bridge network as the peers
    # https://docs.docker.com/compose/networking/
    - CORE_VM_DOCKER_HOSTCONFIG_NETWORKMODE=bal_case-study
    - FABRIC_LOGGING_SPEC=DEBUG
    - CORE_LOGGING_LEVEL=DEBUG
    - CORE_PEER_TLS_ENABLED=true
    - CORE_PEER_PROFILE_ENABLED=true
    - CORE_PEER_TLS_CERT_FILE=/etc/hyperledger/fabric/tls/server.crt
    - CORE_PEER_TLS_KEY_FILE=/etc/hyperledger/fabric/tls/server.key
    - CORE_PEER_TLS_ROOTCERT_FILE=/etc/hyperledger/fabric/tls/ca.crt
    # combine inherited properties as extend is not supported in v3+
    - CORE_PEER_ID=peer0.org1.example.com
    - CORE_PEER_ADDRESS=peer0.org1.example.com:7051
    - CORE_PEER_LISTENADDRESS=0.0.0.0:7051
    - CORE_PEER_CHAINCODEADDRESS=peer0.org1.example.com:7052
    - CORE_PEER_CHAINCODELISTENADDRESS=0.0.0.0:7052
    - CORE_PEER_GOSSIP_EXTERNALENDPOINT=peer0.org1.example.com:7051
    - CORE_PEER_LOCALMSPID=Org1MSP
    - CORE_LEDGER_STATE_STATEDATABASE=CouchDB
    - CORE_LEDGER_STATE_COUCHDBCONFIG_COUCHDBADDRESS=couchdb0:5984
    # The CORE_LEDGER_STATE_COUCHDBCONFIG_USERNAME and CORE_LEDGER_STATE_COUCHDBCONFIG_PASSWORD
    # provide the credentials for ledger to connect to CouchDB.  The username and password must
    # match the username and password set for the associated CouchDB.
    - CORE_LEDGER_STATE_COUCHDBCONFIG_USERNAME=admin
    - CORE_LEDGER_STATE_COUCHDBCONFIG_PASSWORD=adminpw
  working_dir: /opt/gopath/src/github.com/hyperledger/fabric/peer
  command: peer node start
  volumes:
      - /var/run/:/host/var/run/
      - ./fabric/crypto-config/peerOrganizations/org1.example.com/peers/peer0.org1.example.com/msp:/etc/hyperledger/fabric/msp
      - ./fabric/crypto-config/peerOrganizations/org1.example.com/peers/peer0.org1.example.com/tls:/etc/hyperledger/fabric/tls
      - peer0.org1.example.com:/var/hyperledger/production
  depends_on:
    - couchdb0
  ports:
    - 7051:7051
  networks:
    case-study:
      ipv4_address: 172.16.238.14
```

FIGURE 4.5 – Extrait de fichier Docker-compose

La figure 4.5 montre un extrait de fichier de configuration Docker-compose.yaml de la même façon en peut ajouter les autres composants. Ce fichier comporte un entête, les conteneurs d'autorité de certification, les conteneurs de pairs, le conteneur de commande et de CLI (Command Line Interface). Pour créer notre réseau nous utilisons ce fichier, la figure 4.6 décrit chaque composant dans le docker ainsi que son image, ID, sa commande et son port.

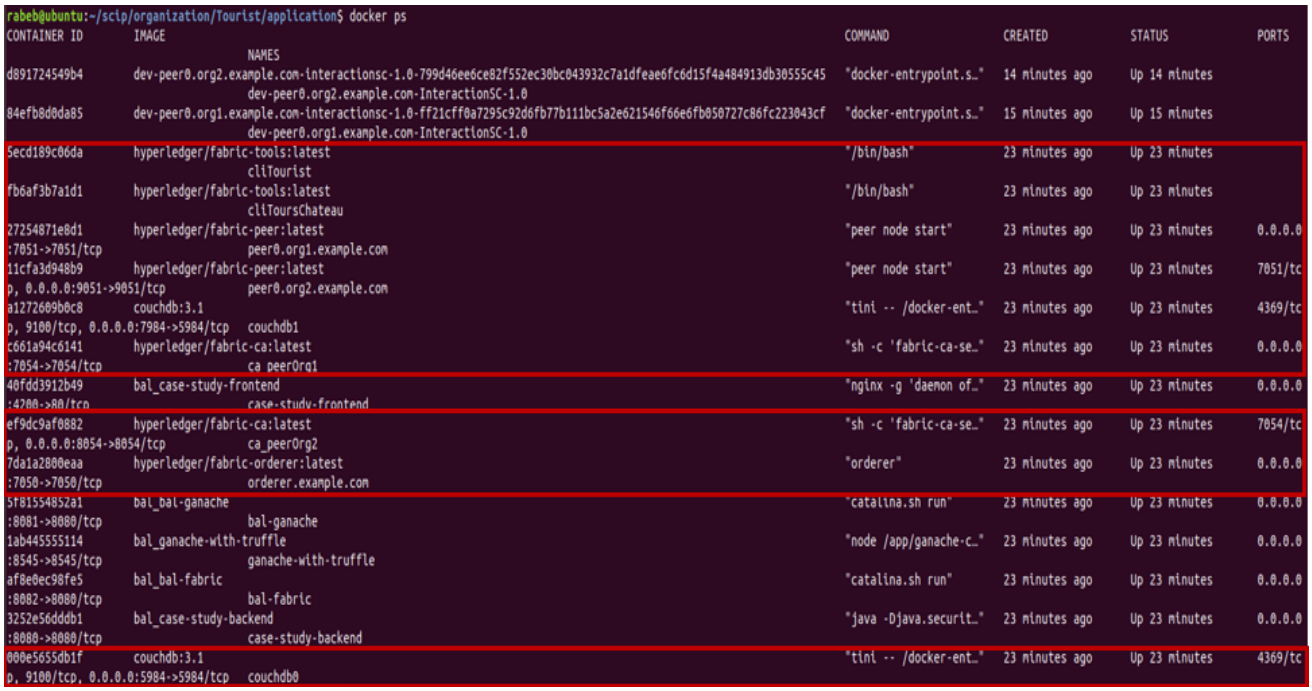


FIGURE 4.6 – Réseau de scénario

Ces conteneurs forment tous un réseau Docker. Dans notre exemple est appelé **bal_case-study**. Chaque conteneur utilise une adresse IP différente, tout en faisant partie d'un seul réseau Docker.

– Les opération de channel

Une fois tous les conteneurs de réseau sont déployés nous devons joindre tous les différents composants au réseau tels que :

- Créer le channel.
- Rejoignez tous les pairs au channel.
- Mettre à jour les anchorpeer.
- Installation et instanciation de chain code

– Déploiement de chain code sur le channel

Dans cette étape, il y a des commandes pour installer et instancier le chaincode lié à l'application de buy/sell ticket sur les pairs.

1. Installer chaincode

Avant que le smart contract InteractionSC puisse être appelé par les applications, il doit être installé sur les noeuds pairs appropriés dans TicketNet. Les administrateurs de ToursChateau et le Touriste peuvent installer le smart contract sur les pairs sur lesquels ils ont respectivement l'autorité. La figure 4.7 illustre l'administrateur de Château de Tours utilise la commande d'installation de pair chaincode pour copier le smart contract InteractionSC de système des fichiers de machine locale vers le système des fichiers dans le conteneur Docker de pair cible.

Une fois que le smart contract est installé sur le pair et instancié sur un channel, peut être invoqué par les applications, et d'interagir avec la base de données de registre via le putState() et getState() Fabric API.

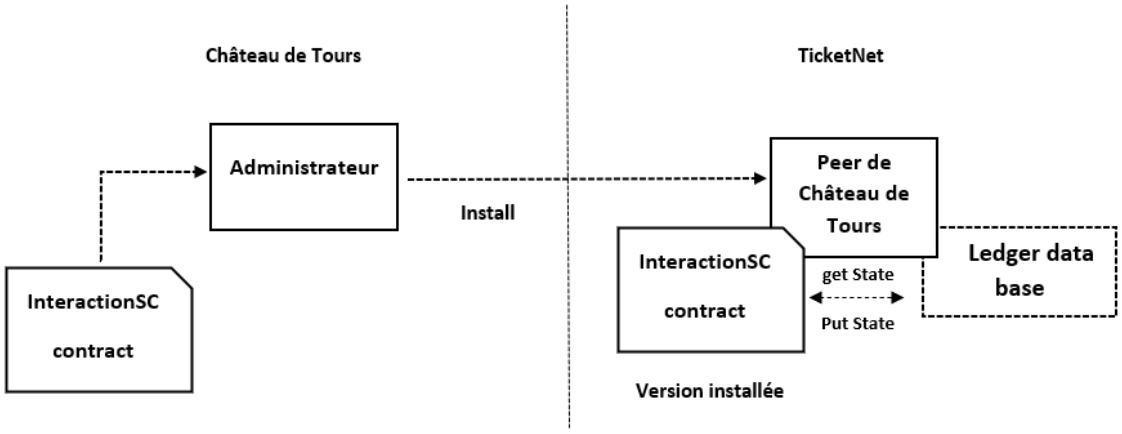


FIGURE 4.7 – Installation de chaincode

2. Instancier chaincode

Après que le chaincode du InteractionSC est installé sur les pairs, un administrateur peut le rendre disponible sur différents canaux du réseau, afin qu'il puisse être appelé par les applications connectées à ces canaux.
La figure 4.8 présente l'administrateur de ToursChateau instancie le chaincode du InteractionSC contenant le smart contract. Un nouveau conteneur de chain code docker est créé pour exécuter InteractionSC.

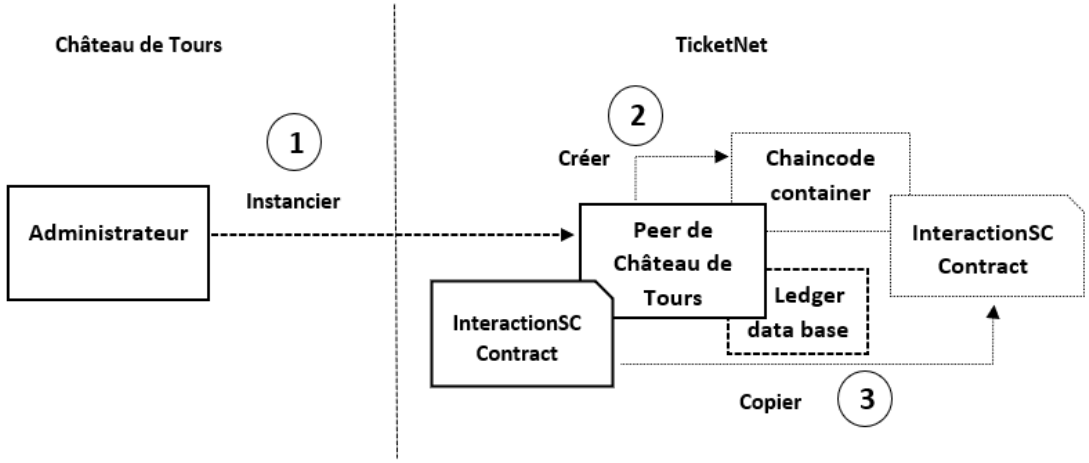


FIGURE 4.8 – Instanciation de chaincode

Le conteneur de chaincode est démarré une fois que l'instance de chaincode a été validée dans le channel. La figure 4.9 présente le conteneur de InteractionSC démarrer sur les deux pairs. Maintenant après avoir déployé le chaincode du InteractionSC sur le channel, nous pouvons utiliser l'application de ToursChateau pour émettre le ticket combiné et l'application de Tourist pour l'acheter.

```
rabeb@ubuntu:~/scip/AutomateSetup$ docker ps
CONTAINER ID        IMAGE                                                                                                                                  COMMAND                  CREATED             STAT
US              PORTS                       NAMES
01dce02a8d91        dev-peer0.org2.example.com-interactionsc-1.0-799d46ee6ce82f552ec30bc043932c7a1dfeae6fc6d15f4a484913db30555c45   "docker-entrypoint.s…"   43 seconds ago      Up 4
1 seconds                                   dev-peer0.org2.example.com-InteractionSC-1.0
3fb4b6290944        dev-peer0.org1.example.com-interactionsc-1.0-ff21cff0a7295c92d6fb77b111bc5a2e621546f66e6fb050727c86fc223043cf   "docker-entrypoint.s…"   47 seconds ago      Up 4
4 seconds                                   dev-peer0.org1.example.com-InteractionSC-1.0
```

FIGURE 4.9 – Conteneur d'instanciation de chaincode

Etape2 : Structure d'application

Le smart contract InteractionSC est appelé par l'application issue.js de ToursChateau. Isabella utilise cette application pour soumettre une transaction au grand livre (Ledger) qui émet le CombinedTicket. Le SDK Fabric permet à une application de se concentrer sur la génération, la soumission et la réponse des transactions. Il coordonne le traitement des propositions de transaction, des orderers et des notifications entre les différents composants du réseau.
Prenons l'exemple de structure de l'application côté de château de tours dans la figure 4.10

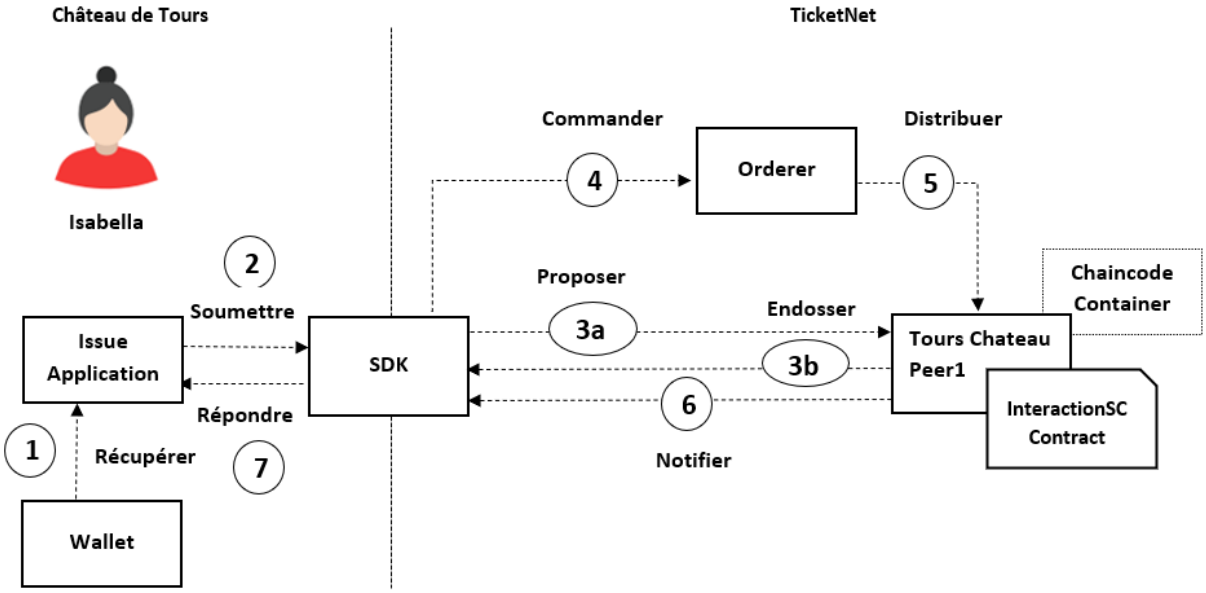


FIGURE 4.10 – Structure d'application

Étant donné que l'application d'émission soumet des transactions au nom d'Isabella, donc elle commence par récupérer le certificat d'Isabella de son portefeuille(Wallet), qui peut être stocké sur le système de fichier local. Puis elle utilise le SDK pour soumettre les transactions sur le channel. Le pair (ToursChateau) simule la transaction proposée, sans mettre à jour le grand livre et l'ordre s'effectue sur le réseau, en parallèle. Il prend les transactions approuvées, classe ces informations dans un bloc et le distribue le bloc à tous les pairs qui ont rejoint le channel, permettant à tout pairs d'exécuter le chaincode dans son propre conteneur de chain code isolé. Enfin, les pairs de validation notifient de manière asynchrone à l'application client le succès ou l'échec de la transaction. Les candidatures seront notifiées par chaque pair qui s'engage.

– Application de Château de tours (issue ticket)

Dans l'application de Tours Château on va exécuter en premier le programme qu'Isabella va utiliser pour charger son identité dans son portefeuille. En effet Elle peut stocker plusieurs identités, dans notre exemple, elle n'en utilise qu'une seule. Le dossier wallet généré contient un fichier isabella.id qui fournit les informations dont Isabella a besoin pour se connecter au réseau comme illustré dans la figure suivante 4.11.

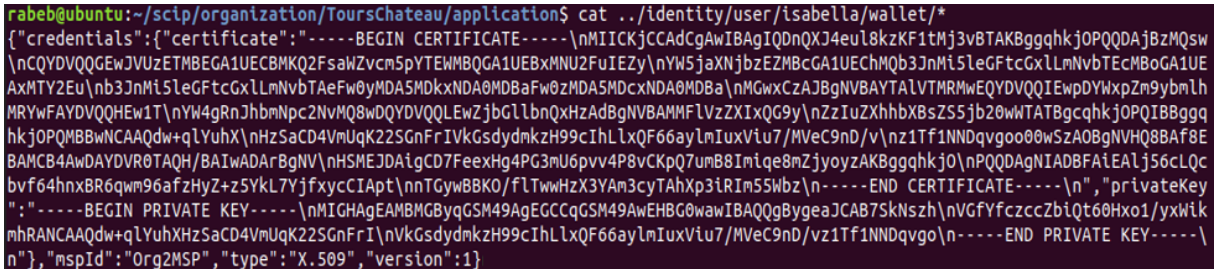


FIGURE 4.11 – Wallet Isabella

- Une **"certificat" :** qui contient la clé publique d'Isabella et d'autres attributs X.509 ajoutés par l'autorité de certification lors de la création du certificat. Ce certificat est distribué sur le réseau afin que différents acteurs à différents moments puissent vérifier cryptographiquement les informations créées par la clé privée d'Isabella.
- Un **"clé privée" :** utilisée pour signer des transactions au nom d'Isabella, mais non distribuée en dehors de son contrôle immédiat.

Après la création d'identité l'application d'issue l'utilise pour créer le ticket combiné au nom de ToursChateau en invoquant le smart contract InteractionSC comme présenté dans la figure 4.12.

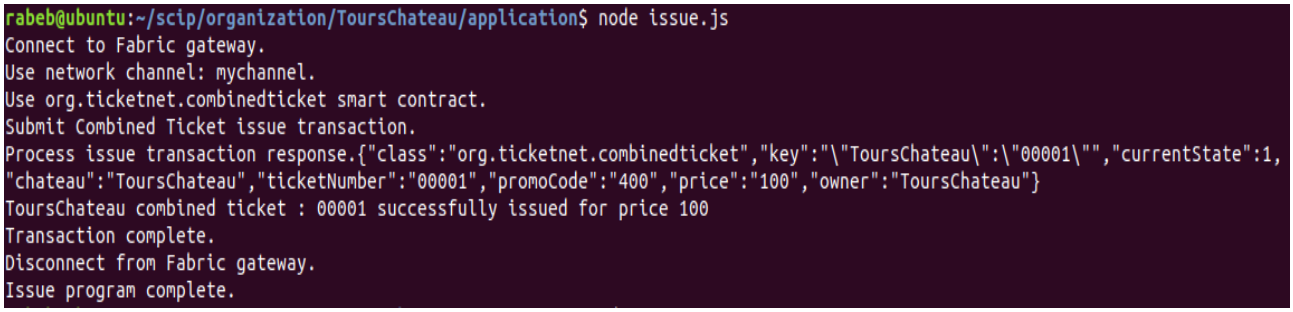


FIGURE 4.12 – Vente de ticket combiné

– Application de Touriste (buy ticket)

Maintenant que le ticket combiné a été émis par ToursChateau, Pour interagir avec TicketNet en tant que tourist. Tout d'abord, nous agissons en tant qu'administrateur qui crée une console configurée pour interagir avec le réseau. Ensuite, Alice, un utilisateur final, utilisera l'application d'achat pour acheter le ticket combiné.
Comme Isabella, Alice peut ajouter des informations d'identité à son portefeuille, qui seront utilisés par buy.js pour soumettre des transactions à TicketNet.
La figure suivante 4.13 montre les informations d'Alice ainsi que l'exécution d'achat de ticket.

FIGURE 4.13 – Achat de ticket combiné

4.4 Conclusion

Dans ce chapitre, nous avons présenté les plateformes qui nous avons utilisé dans l'implémentation de notre approche ainsi que les scénarios développés. Le chapitre suivant sera une conclusion pour notre approche avec les perspectives qui nous pouvons ajouter pour améliorer notre travail.

Conclusion et perspective

L'objectif de ce travail de mémoire était d'assurer une bonne exécution des processus métiers en smart contract qui répond aux problèmes de fiabilité et de correction des processus métiers. Concrètement, nous nous sommes intéressés à développer une approche qui permet une exécution fiable de processus métiers en blockchain a l'aide de l'aspect transactionnel. Les travaux actuels d'exécution de processus métier en blockchain ne permettent pas de résoudre le problème de gestion d'échec donc nous avons proposé d'utiliser les processus métiers transactionnels dans notre solution, d'une part, les TBP assure un bon niveau de correction des échecs, d'autre part la technologie de blockchain assure une exécution en toute confiance et sécurité.

Travail en cours

Dans ce travail nous avons exploité la technologie de smart contract pour permettre une exécution fiable et décentralisée de processus métier et les processus métiers transactionnels pour gérer les échecs et garantir la correction transactionnelle des exécutions de BP. En premier partie nous avons enrichi notre modèle de processus métier avec les propriétés transactionnelles puis nous avons exécuter ses dépendances transactionnelles entre les activités avec la modélisation des mécanismes de recouvrement en cas d'échec en smart contract sur une blockchain publique sans permission (Ethereum). Nous avons étendu le système de Caterpillar pour l'implémenter notre approche. En deuxième partie nous avons étendu notre solution pour l'exécution de collaboration de TBP dans une blockchain privée avec permission (Hyperledger Fabric). L'avantage de notre travail est qu'il assure une exécution fiable et correcte de point de vue de gestion des échecs.

Perspectives

Nos perspectives de recherche concernent l'exécution des processus métiers transactionnels en blockchain. En effet, sous cette appellation, on trouve à la fois la fiabilité, la sécurité, la confiance, la décentralisation.Par conséquent, nous entrevoyons deux perspectives au travail présenté :
Une première direction concerne une extension de BPMN pour prendre en charge la notation de processus métier transactionnel pour pouvoir gérer les retours en arrière et les recouvrements en avant.
La deuxième perspective vise à implémenter des scénarios plus complexe et de voir la possibilité de généraliser son utilisation à d'autres types d'activités.

Bibliographie

[AB18] S. Verekar-A. Pednekar P. Kamat S. Chatterjee A. Baliga, N. Solanki. Performance characterization of hyperledger fabric. In *Crypto Valley Conference on Blockchain Technology (CVCBT)*, pages 65–74, 2018.

[AB19] Y. Sam-S. Bhiri T. Devogele W. Gaaloul A. Brahem, N. Messai. Blockchain's fame reaches the execution of personalized touristic itineraries. In *2019 IEEE 28th International Conference on Enabling Technologies : Infrastructure for Collaborative Enterprises (WETICE)*. IEEE, 2019.

[Abh19] Gunda Abhishek. Property registration and land record management via blockchains. Master's thesis, DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING INDIAN INSTITUTE OF TECHNOLOGY KANPUR, 2019.

[AC98] M. Rusinkiewicz-D. Woelk A. Cichocki, H.A. Ansari. *Workflow and Process Automation : Concepts and Technology*. The Springer International Series in Engineering and Computer Science. SPRINGER SCIENCE+BUSINESS MEDIA, 1998.

[Ant17] Andreas M. Antonopoulos. *Mastering Bitcoin :PROGRAMMING THE OPEN BLOCKCHAIN*. O'Reilly Media, Inc, 2017.

[AS08] White Ph.D A. Stephen, D. Miers. *BPMN modeling and Reference Guide : Understand and using BPMN*. 2008.

[AZ94] B. Bhargava-O. Bukhres A. Zhang, M. Nodine. Ensuring relaxed atomicity for flexible transactions in multidatabase systems. *ACM SIGMOD*, 1994.

[Bhi05] S. Bhiri. *Approche Transactionnelle pour Assurer des Compositions Fiables de Services Web*. PhD thesis, Université Henri Poincaré - Nancy 1, 2005.

[BM09] AK. Elmagarmid B. Medjahed, M. Ouzzani. *Generalization of ACID Properties*, pages 1221–1222. Springer US, 2009.

[C.P16] C.Prybila. Runtime verification for business processes utilizing the blockchain. Master's thesis, Vienna University of Technology, 2016.

[CP17] C. Hochreiner-I. Weber C. Prybila, S. Schulte. Runtime verification for business processes utilizing the bitcoin blockchain. *Future Generation Computer Systems*, 2017.

[DO14] J. Ousterhout D Ongaro. In search of an understandable consensus algorithm. In *USENIX Annual Technical Conference*, 2014.

[DPD18] S. P. Mohanty E. Kougianos D. Puthal, N. Malik and G. Das. Everything you wanted to know about the blockchain : Its promise, components, processes, and problems. *IEEE Consumer Electronics Magazine*, pages 6–14, 2018.

[Dre17] Daniel Drescher. *BLOCKCHAIN BASICS A NON-TECHNICAL INTRODUCTION IN 25 STEPS*. Apress Media LLC, 2017.

[EA18] V. Bortnikov C. Cachin K. Christidis A. De Caro D. Enyeart et al. E. Androulaki, A. Barger. Hyperledger fabric : A distributed operating system for permissioned blockchains. In *EuroSys18 : Thirteenth EuroSys Conference*, number 30, pages 1–15, 2018.

[Elm92] *Database transaction models for advanced applications*. Jim Gray. Morgan kaufmann, 1992.

[GF19] U. Breitenbücher F. Leymann V. Yussupov G. Falazi, M. Hahn. Process-based composition of permissioned and permissionless blockchain smart contracts. In *2019 IEEE 23rd International Enterprise Distributed Object Computing Conference (EDOC)*, pages 77–87, 2019.

[GFL20] F. Daniel A. Lamparelli G. Falazi, U. Breitenbücher and V. Yussupov Leymann, F. Leymann. Smart contract invocation protocol (scip) : A protocol for the uniform integration of heterogeneous blockchain smart contracts. In *Advanced Information Systems Engineering*, pages 134–149. Springer International Publishing, 2020.

[Gro16] BitFury Group, editor. *Public versus Private Blockchains Part 1 : Permissioned Blockchains (White Paper)*, 2016.

[Gup17] Manav Gupta. *Blockchain dummies*. John Wiley & Sons, Inc., ibm limited edition edition, 2017.

[GW92] H.-J. Schek G. Weikum. Concepts and applications of multilevel transactions and open nested transactions. 1992.

[HGM87] K. Salem H. Garcia-Molina. Saga. *ACM*, 1987.

[HSS02] C. Beeri H. Schuldt, G. Alonso and H.-J. Schek. Atomicity and isolation for transactional processes. *ACM Transactions on Database Systems (TODS)*, 2002.

[IW16] R. Riveret G. Governatori A. Ponomarev J. Mendling I. Weber, X. Xu. Untrusted business process monitoring and execution using blockchain. In *International Conference on Business Process Management*, 2016.

[JE95] W. Liebhart J. Eder. The workflow activity model wamo. In *CoopIS*, 1995.

[JEF12] O. K. Takai C. Pu J. E. Ferreira, K. R. Braghetto. Transactional recovery support for robust exception handling in business process services. In *2012 IEEE 19th International Conference on Web Services*, pages 303–310, 2012.

[JL19] I.Weber J. Ladleif, M.Weske. Modeling and enforcing blockchain-based choreographies. In *Business Process Management*, pages 69–85. Springer International Publishing, 2019.

[JM18] W. V. D. Aalst J. V. Brocke C. Cabanillas F. Daniel S. Debois C. D. Ciccio M. Dumas S. Dustdar et al. J. Mendling, I. Weber. Blockchains for business process managementchallenges and opportunities. *ACM Transactions on Management Information Systems (TMIS)*, 9(1) :4, 2018.

[JS97] K. Larry J. Sushil. *Advanced Transaction Models and Architectures*. Springer Science+Business Media, 1997.

[KF18] D. Swanson K. Francisco. The supply chain has no clothes : Technology adoption of blockchain for supply chain transparency. In *Logistics*, number 1, 2018.

[KHE13] MO. KHERBOUCHE. Contributions à la gestion de l'évolution des processus métier. 2013.

[KW18] A. Gervais K. Wust. Do you need a blockchain ? In *Crypto Valley Conference on Blockchain Technology (CVCBT)*, pages 45–54, 2018.

[LA11] S. Cesare L. Aldin. A literature review on business process modelling : New frontiers of reusability. *Entreprise Inform. Syst*, pages 359–383, 2011.

[LA18] A. Anwar L. Amdah. Bpmn profile for collaborative business process. *IEEE 5th International Congress on Information Science and Technology (CiSt)*, pages 42–47, 2018.

[LGBW17] M. Dumas L. Garcia-Banuelos, A. Ponomarev and I. Weber. Optimized execution of business processes on blockchain. In *International Conference on Business Process Management*. Springer, 2017.

[LIC17] L. Tzu-Chun L. Iuon-Chang. A survey of blockchain security issues and challenges. *International Journal of Network Security*, 19(5) :653–659, 2017.

[MD05] WVD. Aalst ; AT. Hofstede M. Dumas. *Process-aware information systems : bridging people and software through process technology*. JWiley-Interscience, 2005.

[MD18] J. Mendling A. Reijers M. Dumas, L. Marcello. *Fundamentals of Business Process Management*. Springer Berlin Heidelberg, 2nd ed edition, 2018.

[MD20] J. Xiao H. Yang X. Ma M. Du, Q. Chen. Supply chain finance innovation using blockchain. *IEEE Transactions on Engineering Management*, 67(4) :1045–1058, 2020.

[MG12] C. Pu E. Ferreira M. Garcia, R. Braghetto. An implementation of a transaction model for business process systems. *Journal of Information and Data Management,*, 3(2) :271–286, 2012.

[Mou16] William Mougayar. *THE BUSINESS BLOCKCHAIN Promise, Practice, and Application of the Next Internet Technology*. John Wiley & Sons, Incorporated., 2016.

[MR94] A. Sheth M. Rusinkiewicz. Specification and execution of transactional workflows. In *IN MODERN DATABASE SYSTEMS : THE OBJECT MODEL, INTEROPERABILITY, AND*. Addison-Wesley/ ACM Press, 1994.

[OLP18] M. Dumas O. Lopez-Pintado, L. Garcia-Banuelos. Business process execution on blockchain. In *CEUR Workshop Proceedings*, volume 2114, 2018.

[OLP19] M. Dumas I. Weber A. Ponomarev O. Lopez-Pintado, L. Garcia-Banuelos. Caterpillar : A business process execution engine on the ethereum blockchain. *Software : Practice and Experience*, 2019.

[Pin20] Orlenys Lopez Pintado. *Collaborative Business Process Execution on the Blockchain : The Caterpillar System*. PhD thesis, DISSERTATIONES INFORMATICAE UNIVERSITATIS TARTUENSIS, 2020.

[PS18] M. Sethumadhavan P. Sajana, M. Sindhu. On blockchain applications : Hyperledger fabric and ethereum. volume 118, pages 2965–2970, 2018.

[SA16] R. Monfared S. Abeyratne. Blockchain ready manufacturing supply chain using distributed ledger. *International Journal of Research in Engineering and Technology*, 05 :1–10, 2016.

[SA17] S. Adarsh S. Asharaf. *Decentralized Computing Using Blockchain Technologies and Smart Contracts*. s published in the IGI Global book series Advances in Information Security, Privacy, 2017.

[SB11] C. Godart O. Perrin M. Zaremba W. Derguech S. Bhiri, W. Gaaloul. Ensuring customised transactional reliability of composite services. *Journal of Database Management(JDM)*, 2011.

[SB18] S. Nadjm-Tehrani S. Bergman, M. Asplund. Permissioned blockchains and distributed databases : A performance study. *Practice and Experience, 2020 - Wiley Online Library*, 2018.

[SK19] H. Arha S. Kamble, A. Gunasekaran. Understanding the blockchain technology adoption in supply chains-indian context. *International Journal of Production*, 57 :2009–2033, 2019.

[SW18] X. Wang-J. Li R. Qin F. Wang S. Wang, Y. Yuan. An overview of smart contract : Architecture, applications, and future trends. In *2018 IEEE Intelligent Vehicles Symposium (IV)*, pages 108–113, 2018.

[Sza94] Nick Szabo. The idea of smart contracts. 1994.

[TW08] Â · B. Kratz P. Grefen T. Wang, J. Vonk. A survey on the history of transaction management : from flat to grid transactions. *Distributed and Parallel Databases*, (23) :253–270, 2008.

[VDA03] Ter Hofstede A. H. Weske M. Van Der Aalst, W. M. Business process management : A survey. Springer, Berlin, Heidelberg, 2003.

[Wes19] Mathias Weske. *Business Process Management Concepts, Languages, Architectures*. Springer, Berlin, Heidelberg, 2019.

[WG09] S. Bhiri A. Haller Ma. Hauswirth W. Gaaloul, K. Gaaloul. Log-based transactional workflow mining. *Distributed and Parallel Databases*, 25(3) :193–240, 2009.

[WG18] W. Yu W. Gao, W. G. Hatcher. A survey of blockchain : Techniques, applications, and challenges. In *2018 27th International Conference on Computer Communication and Networks (ICCCN)*, pages 1–11, 2018.

[WH19] Y. Gao W. Hu, Z. Fan. Research on smart contract optimization method on blockchain. *IT Professional*, 21(5) :33–38, 2019.

[Woo14] Gavin. Wood. Ethereum : A secure decentralised generalized transaction ledger. *Ethereum Project Yellow Paper*, 2014.

[WS97] D. Worah and A. P. Sheth. Transactions in transactional workflows. *In Advanced Transaction Models and Architectures*, pages 3–47, 1997.

[WWJ93] L. Ness A. Sheth W. Woody Jin, M. Rusinkiewicz. Concurrency control and recovery of multidatabase work flows in telecommunication applications. In Peter Buneman and Sushil Jajodia, editors, *Proceedings of the 1993 ACM SIGMOD International Conference on Management of Data, Washington*, pages 456–459. ACM Press, 1993.

[YL92] N. Boudriga Y. Leu, A. Elmagarmid. Specification and execution of transactions for advanced database applications. *Information Systems*, pages 171–183, 1992.

[ZZ17] HN. Dai X. Chen H. Wang Z. Zheng, S. Xie. An overview of blockchain technology : Architecture, consensus, and future trends. *IEEE 6th International Congress on Big Data*, pages 557–564, 2017.