Interopérabilité des Systèmes Blockchains

Léo Besançon

N°d'ordre NNT : 2021LYSE1335

THESE de DOCTORAT DE L'UNIVERSITE DE LYON
opérée au sein de
l'Université Claude Bernard Lyon 1

Ecole Doctorale N° 512
InfoMaths

Spécialité de doctorat : Informatique

Soutenue publiquement le 15/12/2021, par :
Léo Besançon

Interopérabilité des Systèmes Blockchains

Devant le jury composé de :

M. AIT-AMEUR, Yamine	Professeur INPT-ENSEEIHT	Rapporteur
M. CHAROY, François	Professeur Université de Lorraine	Rapporteur
M. EXPOSITO, Ernesto	Professeur UPPA	Président du jury
M. FIGAY, Nicolas	Expert interopérabilité HDR Airbus D&S	Examinateur
Mme HASSAS, Salima	Professeure UCBL1	Examinatrice
Mme GHODOUS, Parisa	Professeure UCBL1	Directrice de thèse
Mme FERREIRA DA SILVA, Catarina	Maître de conférence HDR ISCTE-IUL	Co-directrice de thèse
M. GELAS, Jean-Patrick	Maître de conférence UCBL1	Co-directeur de thèse
M. BURGEL, Éric	Président de B2Expand SAS	Invité

Université Claude Bernard – LYON 1

Président de l'Université	M. Frédéric FLEURY
Président du Conseil Académique	M. Hamda BEN HADID
Vice-Président du Conseil d'Administration	M. Didier REVEL
Vice-Président du Conseil des Etudes et de la Vie Universitaire	M. Philippe CHEVALLIER
Vice-Président de la Commission de Recherche	M. Petru MIRONESCU
Directeur Général des Services	M. Pierre ROLLAND

COMPOSANTES SANTE

Département de Formation et Centre de Recherche en Biologie Humaine	Directrice : Mme Anne-Marie SCHOTT
Faculté d'Odontologie	Doyenne : Mme Dominique SEUX
Faculté de Médecine et Maïeutique Lyon Sud - Charles Mérieux	Doyenne : Mme Carole BURILLON
Faculté de Médecine Lyon-Est	Doyen : M. Gilles RODE
Institut des Sciences et Techniques de la Réadaptation (ISTR)	Directeur : M. Xavier PERROT
Institut des Sciences Pharmaceutiques et Biologiques (ISBP)	Directrice : Mme Christine VINCIGUERRA

COMPOSANTES & DEPARTEMENTS DE SCIENCES & TECHNOLOGIE

Département Génie Electrique et des Procédés (GEP)	Directrice : Mme Rosaria FERRIGNO
Département Informatique	Directeur : M. Behzad SHARIAT
Département Mécanique	Directeur M. Marc BUFFAT
Ecole Supérieure de Chimie, Physique, Electronique (CPE Lyon)	Directeur : Gérard PIGNAULT
Institut de Science Financière et d'Assurances (ISFA)	Directeur : M. Nicolas LEBOISNE
Institut National du Professorat et de l'Education	Administrateur Provisoire : M. Pierre CHAREYRON
Institut Universitaire de Technologie de Lyon 1	Directeur : M. Christophe VITON
Observatoire de Lyon	Directrice : Mme Isabelle DANIEL
Polytechnique Lyon	Directeur : Emmanuel PERRIN
UFR Biosciences	Administratrice provisoire : Mme Kathrin GIESELER
UFR des Sciences et Techniques des Activités Physiques et Sportives (STAPS)	Directeur : M. Yannick VANPOULLE
UFR Faculté des Sciences	Directeur : M. Bruno ANDRIOLETTI

Remerciements

L'élaboration de ce mémoire a été possible grâce à de nombreuses personnes, que je tenais à remercier. Tout d'abord, je remercie vivement mes encadrants de thèse, Madame Parisa GHODOUS, Madame Catarina FERREIRA DA SILVA, ainsi que Monsieur Jean-Patrick GELAS. Ils m'ont apporté un soutien inconditionnel qui m'a permis d'arriver au bout de mes recherches. Leurs expériences et compétences m'ont grandement guidé lors de mes recherches, autant pour l'organisation de mes travaux que pour le travail de fond.

Cette thèse a été réalisée en collaboration avec l'entreprise B2Expand, dans le cadre d'une convention CIFRE. Je tiens ainsi à remercier Monsieur Éric BURGEL, président de B2Expand, qui m'a permis d'obtenir le financement de cette thèse, et de m'avoir fait confiance afin d'appliquer les résultats de recherche obtenus dans un cadre industriel innovant.

Je remercie les membres du comité de suivi de ma thèse, Monsieur Ernesto EXPOSITO et Madame Sylvie CALABRETTO qui ont pu garder un œil expert sur l'avancement et l'évolution de ma thèse.

De nombreuses autres personnes m'ont apporté leur aide, de près ou de loin. Je souhaitais citer par exemple Monsieur François GUÉZENGAR et Monsieur Nicolas FIGAY, avec qui j'ai pu avoir des échanges techniques intéressants, ainsi que l'association CryptoLyon avec qui j'ai pu échanger sur de nombreux sujets.

Dans un contexte plus personnel, je tiens à remercier mes proches qui ont pu m'aider lors de moments difficiles. En particulier, je remercie Nghi pour son soutien quotidien, sans qui je n'aurai pas pu aborder l'écriture de ce manuscrit sereinement. Je remercie également ma famille, ainsi que mes amis, entre bien d'autres Florent, Simon et Rémi, pour leur soutien.

Durant ces deux dernières années, la situation sanitaire a fortement impacté nos conditions de travail et de vie. Je remercie ainsi Céline, qui m'a permis de garder un équilibre psychologique durant cette période. Par ailleurs, j'apporte ma reconnaissance envers Jean-Patrick qui m'a initié au parachutisme, un sport qui permet de s'évader instantanément.

Résumé

La Blockchain est une technologie disruptive. Elle s'intègre dans un écosystème décentralisé d'applications aux propriétés intéressantes : la transparence des transactions, l'auditabilité des applications, ou encore la résistance à la censure. Les domaines d'application sont variés, de la finance à la santé ou au jeu vidéo. La technologie a évolué depuis sa création en 2008 et possède de nombreuses perspectives.

Néanmoins, le domaine rencontre de nombreux défis. Chaque Blockchain utilisant ses propres standards et modèles économiques, il subit notamment un manque d'interopérabilité à différents niveaux : entre les différents projets d'une Blockchain, entre les différentes Blockchains, ainsi qu'entre les Blockchains et les autres systèmes.

Un aspect important de l'interopérabilité des systèmes Blockchains est leur interopérabilité sémantique, qui nécessite de définir formellement les concepts liés.

Un autre défi est la conception d'applications Blockchains décentralisées. Ces applications intègrent la technologie Blockchain, mais aussi d'autres services qui permettent de satisfaire les contraintes de l'application pour lesquelles la Blockchain n'est pas adaptée. Cependant, il est complexe de choisir les services Blockchain les plus adaptés à une application donnée.

Cette thèse a pour objectif la proposition d'un cadre permettant d'améliorer l'interopérabilité des applications Blockchain décentralisées. Pour cela, nous développons une méthodologie d'aide à la conception de ces applications, ainsi qu'une ontologie Blockchain qui aide à formaliser leurs concepts.

Ce cadre est validé dans le domaine des jeux vidéo Blockchain. Cet environnement est complexe, car il nécessite le partage de données volumineuses. De plus, les contraintes de latence doivent être respectées.

Mots-clés : Blockchain, Interopérabilité, DApps, Ontologie, SWRL, Systèmes distribués

Summary

Title in English: Blockchain Systems Interoperability

Blockchains are a disruptive technology. They enable an ecosystem for decentralized applications with interesting properties: transaction transparency, application auditability or censorship resistance. It has applications in various fields, such as finance, healthcare or video games. It has evolved a lot since its creation in 2008, and presents numerous perspectives.

However, the field faces many challenges, in particular a lack of interoperability at several levels: between projects on a same Blockchain, between different Blockchains, or between Blockchains and other systems.

One important aspect of Blockchain systems interoperability is semantic interoperability. It relies on formal definitions of the related concepts.

Another challenge is the design of decentralized Blockchain applications. These applications integrate Blockchain technology, but also other services that satisfy the constraints of the application that Blockchain is not suitable for. However, it is complex to choose the most suited Blockchain service for a given application.

With this PhD work, we propose a framework that can improve interoperability for decentralized Blockchain applications. We achieve this with the design of a methodology and a Blockchain ontology that help formalize the concepts related to an application.

This framework is validated in the context of Blockchain video game development. It is a complex use case, as it needs to share storage intensive data and satisfy the latency constraints.

Keywords: Blockchain, Interoperability, DApps, Ontologies, SWRL, Distributed Systems

Table des matières

Liste des Acronymes

AOWLN Aided OWL Notation. 115, 116, 119, 124

API Application Programming Interface. 40, 61, 107, 155

AWS Amazon Web Services. 27

BLONDiE BLockchain ONtology with Dynamic Extensibility. 12, 47, 56, 65–67, 88, 89

BPMN Business Process Model and Notation. 6, 12, 14, 19, 49–51, 60, 81, 82, 84, 86, 137, 138, 146, 147, 153, 154, 156, 158

DAO Decentralized Autonomous Organization. 13, 35, 105–107

DApp Decentralized Application. 7, 12, 13, 15, 16, 48, 49, 51, 60, 61, 64–67, 76, 78–87, 89–91, 93–103, 105, 107–110, 112, 113, 115, 121–130, 136, 138, 146, 147, 150–158, 160

DApps Decentralized Applications. 6, 29, 30, 48, 49, 60–64, 66, 67, 70, 77, 79–81, 84, 87, 88, 93, 98, 102, 103, 105, 108, 109, 112, 121, 131, 132, 134, 150, 151, 153, 159, 161

DBA Decentralized Blockchain Applications. 87, 121

DL Description Logics. 41

DLT Distributed Ledger Technologies. 12, 34, 62, 63

DPoS Delegated Proof of Stake. 32, 72, 74, 80, 93, 130, 145

EEA Enterprise Ethereum Alliance. 59

EIP Ethereum Improvement Proposal. 58

ERC Ethereum Request for Comment. 58, 59, 85, 99, 100, 102, 140, 141, 143, 144

EthOn Ethereum Ontology. 12, 47, 56, 65–67, 82, 88, 89, 112

EVM Ethereum Virtual Machine. 59, 76, 143

IBM International Business Machines Corporation. 34, 64

IEEE Institute of Electrical and Electronics Engineers. 58–60

IoT Internet of Things. 64, 65, 67

IPFS InterPlanetary File System. 13, 61, 74, 75, 81, 86, 97

LCIM Levels of Conceptual Interoperability Model. 38, 39

LISI Levels of Information Systems Interoperability. 38

LTR Light Trail Rush. 8, 13, 14, 22, 23, 97, 98, 134, 139, 140, 142, 143, 146–148, 152, 155–158

NFT Non-Fongible Token. 7, 13, 57, 97, 102–105, 110, 112, 134, 140, 146, 155, 157, 158

NIST National Institute of Standards and Technology. 25, 58, 59, 91

NPoS Nominated Proof of Stake. 74

OWA Open-World Assumption. 128

OWL Ontology Web Language. 23, 41, 56, 65, 83, 114, 122, 128, 131, 132

PoA Proof of Authority. 32, 72, 92, 93

PoS Proof of Stake. 30, 32, 72–74, 80, 93, 145, 155

PoW Proof of Work. 30–32, 56, 57, 80, 92, 93, 96, 116, 117, 130, 145

REST REpresentational State Transfer. 40

SDK Software Development Kit. 21, 61

SMTP Simple Mail Transfer Protocol. 40

SOA Service-Oriented Architecture. 12, 62

SPA Single-Page Application. 81, 86, 97

SWRL Semantic Web Rule Language. 7, 16, 23, 93, 96, 97, 99, 112–114, 119–122, 128–132, 145, 151, 152, 156, 159, 160

TCP Transmission Control Protocol. 27, 65

TEE Trusted Execution Environment. 64

UDP User Datagram Protocol. 27, 65

UML Unified Modeling Language. 48

UTXO Unspent Transaction Output. 66, 82

ZK-rollups Zero-Knowledge Rollups. 68, 70–72, 78, 94

ZKP Zero-Knowledge Proofs. 70, 71, 77, 78

Table des figures

Liste des tableaux

Liste des listings

Introduction

Présentation du sujet

L'informatique est un domaine de recherche relativement nouveau. Il subit ainsi encore de nombreuses innovations, dans des domaines variés tels que l'intelligence artificielle, le traitement d'image, ou encore la cryptographie, le calcul distribué et les bases de données distribuées.

Ces trois derniers domaines de l'informatique se sont développés ces dernières décennies, permettant l'apparition d'une nouvelle technologie à la fin des années 2000 : la technologie Blockchain, qui permet non plus un simple transfert d'informations à travers un réseau, mais également un transfert de valeur de façon décentralisée, sans nécessiter un tiers de confiance [1].

Le principe de base de la technologie est de permettre à un réseau pair-à-pair de modifier un registre de compte commun à tous les nœuds du réseau. Il s'agit donc d'une structure de données distribuée et modifiable par tous. Pour ce faire, les différents nœuds peuvent signer des messages cryptographiques attestant d'une transaction, et le protocole de la Blockchain assure le consensus décentralisé des modifications réalisées.

Depuis son apparition, de nombreux changements ont amélioré cette technologie. Les applications utilisant la Blockchain sont dorénavant davantage financées, le nombre d'acteurs croît considérablement, et les possibilités qu'elles offrent se multiplient.

Cependant, cet élargissement de l'écosystème des Blockchains amène des problématiques de recherches diverses.

Problématique

Pour guider nos travaux, nous cherchons à répondre à un certain nombre de questions de recherche relatives au sujet présenté précédemment.

Tout d'abord, nous devons nous poser la question suivante.

Question de recherche 1 : Que signifie l'interopérabilité appliquée aux systèmes Blockchains, quelles en sont les motivations, et en quoi est-elle limitée actuellement ?

Cette question est importante puisqu'elle permet de contextualiser le sujet et en définir les limites. Nous verrons ainsi que l'interopérabilité peut se diviser en plusieurs niveaux au sein de l'écosystème Blockchain. Nous allons alors focaliser nos travaux sur un de ces niveaux.

Une fois les concepts de base bien compris, il nous faut résoudre un point précis de l'interopérabilité des systèmes Blockchains. Nous posons alors la question suivante.

Question de recherche 2 : Comment améliorer l'interopérabilité entre les composants d'une application Blockchain décentralisée ?

Dans cette deuxième question, nous nous focalisons donc sur l'interopérabilité des applications Blockchain décentralisées, qui est une partie importante de l'interopérabilité appliquée au domaine de la Blockchain. Nous cherchons donc à proposer des méthodes permettant d'améliorer cette interopérabilité.

Une fois les contributions améliorant l'interopérabilité au sein des applications Blockchain décentralisées proposées, il nous faut les évaluer par le biais d'une application industrielle. C'est pourquoi nous posons la question suivante.

Question de recherche 3 : Comment mettre en application nos travaux sur l'interopérabilité des systèmes Blockchain dans un cadre industriel ?

Nous devons ainsi valider et évaluer les contributions de cette thèse dans un cadre industriel afin de vérifier que nous répondons bien au problème posé. Ce cadre, concernant le domaine des jeux vidéo Blockchain, est détaillé par la suite.

Ces trois questions nous permettent d'étudier la problématique suivante :

Problématique : « Quels approches et outils permettraient une amélioration de l'interopérabilité au cœur des Applications Blockchain Décentralisées, et quelles sont leurs contraintes ? »

En effet, l'étude de cette problématique nous permet de bien définir les termes associés, et de nous assurer que les contributions proposées servent concrètement à pallier les limitations actuelles de la technologie Blockchain.

Cette problématique montre que les objectifs de cette thèse sont multiples :

- Comprendre les limitations de la technologie Blockchain, et leurs évolutions,

- Proposer un cadre d'étude de l'interopérabilité dans ce contexte,
- Proposer des outils et des approches pour améliorer concrètement l'interopérabilité des systèmes Blockchains,
- Évaluer les contributions proposées par rapport à l'interopérabilité des systèmes Blockchain.

Nous développons dans la section suivante les contributions de nos travaux de recherche qui répondent à ces objectifs.

Contributions

L'approche de nos travaux part du constat initial suivant : il existe un manque d'interopérabilité entre les composants d'une application Blockchain décentralisée. En particulier, de nombreuses technologies liées à la Blockchain existent et apparaissent régulièrement, et cela a pour effet qu'un développeur d'applications Blockchain décentralisées n'a pas de moyen aisé de savoir lesquelles sont le plus adaptées à son cas d'application. Il existe en outre un manque d'interopérabilité sémantique entre ces technologies. Cela conduit à des difficultés lors de la formalisation des applications Blockchains décentralisées, que nous cherchons à résoudre.

Afin de répondre à nos questions de recherches et à notre problématique, nous proposons ainsi dans cette section les deux contributions majeures de nos travaux.

Première contribution

Dans un premier temps, nous établissons une méthodologie améliorant l'interopérabilité des applications Blockchains décentralisées. En effet, cette méthodologie aide les chercheurs et développeurs de telles applications à déterminer quels services sont adaptés à leur application. Ainsi, nous avons commencé par étudier ce dont nous avons besoin pour résoudre ce problème. La formalisation de l'application est alors nécessaire, et nous utilisons différents outils et approches pour réaliser cette formalisation.

La méthodologie que nous proposons contient quatre étapes. Tout d'abord, nous proposons de formaliser les concepts liés à l'application voulue, ainsi que leurs relations. Cela peut être réalisé grâce à une ontologie. Ensuite, il faut modéliser les différentes interactions au sein de l'application. Ces interactions peuvent correspondre aux interactions entre les utilisateurs et l'application, ou bien entre les différents services qui composent l'application. Un outil de modélisation de processus tel que *Business Process Model and Notation* (BPMN) peut être utilisé pour cette étape. Une fois l'application formalisée ainsi, les contraintes de l'application doivent être déduites. Ces contraintes peuvent être fonctionnelles

ou technologiques. Enfin, la dernière étape de la méthodologie consiste à choisir, grâce aux contraintes de l'application, les instances de services Blockchains les plus adaptées à l'application.

Nous avons ainsi abouti à une méthodologie qui utilise ces deux approches pour en déduire quelles instances de services Blockchain le développeur doit utiliser au sein de son application.

Deuxième contribution

Notre première contribution nécessite ainsi la création d'une ontologie permettant de formaliser les concepts d'une application donnée qui intègre la technologie Blockchain. Cependant, nous avons remarqué que la création d'une ontologie formalisant les concepts d'une application n'est pas triviale.

Dans un second temps, nous proposons donc une ontologie générique des systèmes Blockchains qui formalise de nombreux concepts liés aux applications Blockchains décentralisées. Cette ontologie permet de résoudre en partie l'interopérabilité sémantique des systèmes Blockchains.

Cette ontologie générique peut alors servir de base pour modéliser n'importe qu'elle application Blockchain décentralisée. De surcroît, d'autres chercheurs et développeurs peuvent étendre cette ontologie pour facilement formaliser les concepts de leurs applications. Les applications formalisées à partir de notre ontologie seront alors interopérables sémantiquement.

Validation dans le cadre industriel

Finalement, une fois ces deux contributions développées, nous les évaluons au sein d'une application industrielle. Pour cela, le dernier jeu vidéo Blockchain en développement par B2Expand, notre partenaire industriel, a été un cas d'application concret.

Contexte des travaux

Cette étude a été réalisée dans le cadre d'une thèse en contrat CIFRE, la Convention Industrielle de Formation par la REcherche, avec l'entreprise B2Expand, située à Villeurbanne. Ce studio de jeu vidéo indépendant utilise la Blockchain afin d'apporter aux joueurs la possibilité d'échanger facilement leurs assets avec d'autres joueurs.

L'entreprise a réalisé la 1ère levée de fond en cryptomonnaie de France, fin 2016, et a sorti en octobre 2018 son premier jeu, Beyond the Void. La FIGURE 1 montre une image du jeu. Ce jeu de stratégie dans l'espace gratuit propose la vente d'assets Blockchain. Ces assets sont des modèles 3D de vaisseaux spatiaux. Ainsi, si un joueur possède l'asset Blockchain correspondant à un vaisseau, alors

FIG. 1 : Image du premier jeu de B2Expand, Beyond the Void.

il peut l'utiliser au sein du jeu. Cette approche possède plusieurs avantages pour les joueurs. Par exemple, il est plus difficile pour l'entreprise de modifier les caractéristiques d'un asset sans la volonté des joueurs. Elle permet également la mise en place d'un marché secondaire d'assets, c'est-à-dire la revente à d'autres joueurs des assets achetés.

L'objectif de l'entreprise est de réaliser des jeux de plus en plus décentralisés, qui n'utilisent pas seulement la technologie Blockchain pour les objets virtuels des joueurs, mais également pour proposer des mécaniques de jeu innovantes, ainsi que pour rendre leurs jeux davantage décentralisés.

Améliorer l'interopérabilité des systèmes Blockchains apporterait plusieurs avantages pour le développement des jeux vidéo à B2Expand. Un exemple d'outil utile serait de donner la possibilité aux développeurs d'un jeu vidéo Blockchain de changer la Blockchain utilisée sans avoir besoin de changer leur environnement de développement. L'implémentation consisterait alors en des *Software Development Kit* (SDK) et bibliothèques logicielles liées aux moteurs de jeux afin de s'interfacer facilement avec plusieurs Blockchains. À partir de cela, l'entreprise arriverait mieux à comprendre comment mettre à profit la technologie Blockchain au sein de ses jeux vidéo. Par ailleurs, l'interopérabilité des applications Blockchain décentralisées permettrait de faciliter la mise en place de partenariats avec d'autres studios de jeu vidéo Blockchain. Le *cross-gaming* permet aux joueurs d'un jeu qui possède un certain asset d'avoir accès automatiquement à des fonctionnalités additionnelles d'un autre jeu vidéo.

L'utilisation des technologies liées aux applications décentralisées et aux Blockchains au sein d'un jeu vidéo apporte de multiples avantages, comme une plus

grande confiance entre les développeurs du jeu et les joueurs, ou encore la simplification de l'implémentation de certaines fonctionnalités :

- Le créateur d'Ethereum [2], Vitalik Buterin, a réalisé l'importance de la décentralisation quand l'éditeur de jeu Blizzard à unilatéralement mit à jour les règles concernant l'un de ses objets dans le jeu World of Warcraft [3]. Le joueur s'est senti trahi par les développeurs, et il a compris qu'il ne possédait pas réellement ses objets virtuels.
- Si les joueurs utilisent uniquement des technologies pair-à-pair, il n'y a pas besoin de maintenir de serveurs, et les développeurs réduisent leurs coûts.
- L'utilisation de la Blockchain au sein d'un jeu vidéo permet d'utiliser les fonctionnalités de l'écosystème Blockchain, comme les transferts d'argents décentralisés. Nous pouvons ainsi penser à des paris en cryptomonnaie sur l'issue d'une partie.

C'est pourquoi de nombreux jeux Blockchains sont apparus ces dernières années. Le premier, Huntercoin [4], développé en 2012, était particulier puisque le jeu était lui-même le client d'un nœud d'une Blockchain du même nom. Ainsi, les règles du jeu étaient intégrées directement au niveau du protocole du réseau. Depuis, le projet a évolué pour devenir Xaya [5] une plateforme supportant plusieurs jeux.

En outre, l'apparition des contrats intelligents, et plus particulièrement de la Blockchain Ethereum, a permis l'apparition de nombreux autres projets de jeux Blockchain, tels que *Light Trail Rush* (LTR) [6], CryptoKitty [7] ou encore Decentraland [8]. Clack et al. [9] proposent de définir les contrats intelligents comme suit : « Un contrat intelligent est une entente automatisable et applicable. Automatisable par un ordinateur, même si certaines parties peuvent demander des contrôles et gestions humaines. Applicable soit par des droits et obligations, soit via l'exécution inviolable de code informatique. » Cette définition est volontairement très générique, pour convenir à de nombreux scénarios. De manière plus spécifique, nous définissons dans ce manuscrit les contrats intelligents comme des programmes informatiques s'exécutant de manière décentralisée, et sécurisée par des mécanismes de consensus tels que les Blockchains.

Organisation de la thèse

Le premier chapitre permet d'étudier les concepts principaux qui fondent la technologie Blockchain et l'interopérabilité. Nous expliquons ainsi comment les Blockchains permettent les transferts de valeur à travers un réseau décentralisé. Nous décrivons les Blockchains sous trois angles principaux : sous la forme d'un protocole de communication, puis sous la forme d'une structure de données

distribuées, et enfin sous la forme de leurs fonctionnalités, telles que les contrats intelligents. Nous détaillons également les différents types d'interopérabilité des systèmes. À partir de cela, nous répondrons aux deuxième et troisième questions de recherche.

Le deuxième chapitre présente un état de l'art de différents types d'interopérabilité au sein de ce domaine. En effet, l'interopérabilité peut être recherchée au niveau des données ou de la syntaxe, mais également au niveau de la sémantique des concepts du domaine. Nous présentons également les différents services qui permettent aux applications de tirer pleinement parti du potentiel de la technologie. Ces services peuvent aider à la scalabilité du nombre de transactions possibles, à baisser leurs coûts, ou encore au stockage de données distribuées. Cet état de l'art, en complément du premier chapitre, répond alors à la première question de recherche que nous nous sommes posée : « Que signifie l'interopérabilité appliquée aux systèmes Blockchains, quelles en sont les motivations, et en quoi est-elle limitée actuellement ? »

Le troisième chapitre décrit notre première contribution, relative à l'aide à la conception d'applications Blockchain décentralisées. Pour cela, nous présentons une méthodologie d'aide à cette conception, qui demande aux développeurs de telles applications de les formaliser afin d'en déduire les contraintes, puis les services Blockchain adaptés à leur application.

Le quatrième chapitre décrit notre deuxième contribution. Cette contribution consiste en la création d'une ontologie générique servant à formaliser les concepts clés des applications Blockchain décentralisées. Cette ontologie est étendue d'une ontologie Blockchain existante. Elle formalise des concepts Blockchain génériques, mais également des concepts plus spécifiques à certaines applications.

Ensuite, au sein du cinquième chapitre, nous présentons l'utilisation de cette ontologie pour le développement d'applications. En particulier, nous étudions des règles *Semantic Web Rule Language* (SWRL) qui permettent de contraindre les concepts de l'ontologie de manière plus expressive que *Ontology Web Language* (OWL) de celle-ci. Nous analysons également comment formaliser des applications utilisant la Blockchain de différentes manières. Cela nous permet de discuter de différents cas d'applications possibles de nos contributions. Les troisième, quatrième et cinquième chapitres décrivent ainsi nos contributions de recherche, qui répondent à la question « Comment améliorer l'interopérabilité entre les composants d'une application Blockchain décentralisée ? »

Le sixième chapitre présente l'application industrielle de nos recherches, qui consiste en l'utilisation de notre méthodologie et de notre ontologie pour le développement d'un jeu vidéo Blockchain. Nous commençons l'étude sur le cas simple du jeu d'échecs avant d'aborder l'utilisation de la Blockchain dans le jeu *Light Trail Rush* (LTR) de l'entreprise B2Expand. Ce dernier cas d'application

permet de répondre à notre dernière question de recherche : « Comment mettre en application nos travaux sur l'interopérabilité des systèmes Blockchain dans un cadre industriel ? »

Le septième chapitre évalue nos contributions. Nous développons en premier lieu les critères d'évaluations retenus pour nos contributions de recherche et notre application industrielle, avant de présenter les résultats obtenus.

Enfin, nous concluons ce mémoire en discutant de plusieurs perspectives de nos travaux de recherches.

Chapitre 1

Blockchains et Interopérabilité

Ce chapitre présente le contexte de nos recherches, ainsi que les concepts de base liés à nos travaux. Ainsi, nous développons le fonctionnement de la technologie Blockchain et de son écosystème, ainsi que les concepts de base de l'interopérabilité. Finalement, nous présentons les motivations et limitations de l'interopérabilité des systèmes Blockchains.

1.1 Introduction

La technologie Blockchain est née en 2008 avec l'apparition du réseau Bitcoin [10], créé anonymement, sous le pseudonyme de Satoshi Nakamoto. Elle possède des applications dans de nombreux domaines, tels que les chaînes logistiques, ou *supply chain* [11], la finance [12], la santé [13] ou encore les jeux vidéo [5].

D'après le *National Institute of Standards and Technology* (NIST), une Blockchain est un livre de registre collaboratif et résistante aux tentatives de falsifications, qui maintient des données transactionnelles [14]. Dans les Blockchains dites « publiques », cet effort collaboratif s'effectue sans autorité centrale. En revanche, pour les Blockchains dites « privées », les rôles d'utilisation du réseau et de validation des modifications sont asymétriques.

La Blockchain est une technologie récente qui utilise les concepts de nombreux domaines : l'informatique distribuée, les protocoles de communications, la vérification d'informations, ou encore la cryptographie. Cette technologie propose alors des innovations relatives à la confiance que des utilisateurs peuvent s'octroyer mutuellement dans un environnement décentralisé. Comme nous le

décrivons dans ce mémoire, de plus en plus d'applications cherchent à intégrer cette technologie.

Pour certaines applications, comme par exemple l'utilisation de la Blockchain au sein de jeux vidéo [15] cela amène à un besoin d'obtenir un grand nombre d'échanges de données volumineuses avec une latence minimale. Ces besoins semblent complexes à intégrer au sein d'une architecture distribuée utilisant la technologie Blockchain.

Nous commençons par reprendre la terminologie des concepts liés à l'étude, afin de clarifier le contexte de l'étude. Nous définissons ainsi les concepts liés à la technologie Blockchain et à l'interopérabilité. Ensuite, nous étudions le rôle de l'interopérabilité au sein du domaine de la Blockchain. Nous exposons finalement les motivations industrielles de l'étude.

1.2 Technologie Blockchain et applications décentralisées

Ce chapitre vise à définir la terminologie utilisée, contextualiser le domaine d'étude, et analyser les différentes problématiques de recherches liées.

1.2.1 Terminologie et contextualisation de l'étude

La Blockchain est un domaine disruptif pour de nombreuses industries. La terminologie employée par les différents chercheurs et industriels de ce domaine réutilise, et étend, les terminologies employées jusqu'à présent dans d'autres domaines, comme le calcul distribué ou les protocoles décentralisés. Nous détaillons dans cette sous-section cette terminologie employée.

Architectures logicielles

Une infrastructure logicielle est définie par l'ensemble des composants nécessaires afin de concevoir, implémenter, déployer et exécuter une application informatique [16]. Depuis les dernières décennies, les modèles de calculs et les infrastructures logicielles n'ont cessé de se complexifier. En effet, nous sommes passés de paradigmes tels que le calcul parallèle au calcul distribué, puis l'utilisation de grilles informatiques (ou *grid computing*), et enfin, plus récemment, du *cloud computing* [17].

Cette complexification successive des architectures techniques utilisées pour réaliser des calculs a apporté un grand nombre d'avantages, tels que la réplication de données, la tolérance aux pannes, un partage plus efficace des ressources disponibles, une élasticité plus importante ou encore la simplification au déploiement et à l'exécution des tâches informatiques. La sécurité des données est également devenue un sujet important. Cependant, le bon fonctionnement des

systèmes classiques repose sur la confiance que l'on octroie aux composants de ce système.

Aujourd'hui, les architectures logicielles peuvent montrer un grand niveau d'abstraction, via l'utilisation du concept de service. L'Organization for the Advancement of Structured Information Standards (OASIS) définit un service comme un « mécanisme pour permettre l'accès à une ou plusieurs fonctionnalités, où l'accès est fournit par l'utilisation d'une interface spécifique » [18].

Afin de mettre en place des architectures complexes, par exemple fonctionnant sur plusieurs machines, il est également nécessaire d'utiliser des protocoles adaptés. À l'échelle des réseaux de machines, les différents composants nécessaires à faire fonctionner un logiciel à travers un réseau doivent supporter un ou plusieurs protocoles de communication. Ces protocoles, souvent standardisés, régissent les messages que ces composants doivent se transmettre afin d'assurer le bon fonctionnement du réseau.

Parmi les protocoles couramment utilisés, nous pouvons par exemple distinguer les protocoles de communication tels que Hypertext Transfer Protocol (HTTP) (version 1.0 ou 2.0), les requêtes Structured Query Language (SQL), ou encore *Transmission Control Protocol* (TCP)/*User Datagram Protocol* (UDP) qui font partie de la couche de transport du modèle Open Systems Interconnection (OSI). D'autres protocoles peuvent être utilisés, par exemple des protocoles de partage de données, tels que le File Transfer Protocol (FTP) dans un réseau avec une architecture client-serveur ou BitTorrent pour le partage de données en grande partie décentralisé à travers un réseau pair-à-pair.

Applications distribuées et décentralisées

Une différenciation importante à faire est celle entre les notions d'environnements distribués et décentralisés. Parfois vus comme interchangeables, ces termes définissent des concepts bien différents. Une application distribuée répartit différentes tâches à travers plusieurs machines. Les objectifs de la distribution d'une application peuvent alors être la redondance des informations, l'accélération des calculs parallélisables, ou encore la tolérance aux pannes. Il s'agit ainsi d'une notion plus générale que la parallélisation des calculs. L'informatique parallèle n'a pour objectif que l'accélération des calculs via une répartition des tâches sur différents processeurs.

En revanche, une application distribuée peut toujours posséder un contrôle central par une entité. C'est par exemple le cas des services de stockage distribué (Cloud) gérés par Google ou de calcul distribué géré par Amazon avec *Amazon Web Services* (AWS). Ces applications sont généralement tolérantes aux pannes, mais pas forcément tolérantes aux pannes byzantines. Lamport et al. [19] décrivent les pannes byzantines comme étant toute panne d'un nœud d'un système distribué menant à des informations conflictuelles au sein du réseau. Les pannes byzantines arrivent par exemple lorsqu'un système distribué

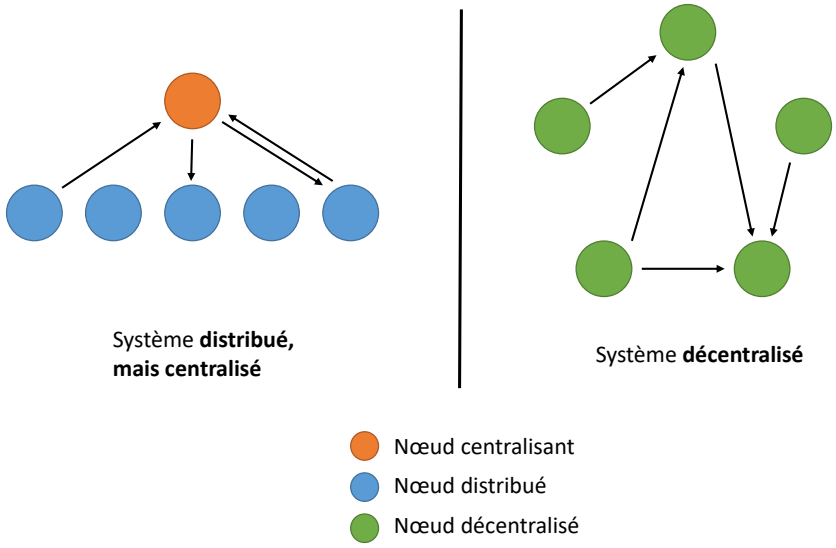

FIG. 1.1 : Représentation graphique d'un réseau distribué (à gauche) et décentralisé (à droite).

n'arrive pas à détecter la panne d'un des nœuds du réseau. Souvent, on définit une panne byzantine comme étant résultante d'une action malicieuse, même si ce n'est pas toujours le cas.

Au contraire, les applications décentralisées n'ont pas de contrôle central. Des exemples de telles applications sont le partage de fichier par le protocole BitTorrent [20], ou encore Bridgefy [21], un réseau social assurant un système de messagerie entre utilisateurs en relayant les messages par technologie Bluetooth. Ainsi, les nœuds du réseau doivent être proches géographiquement afin d'assurer la transmission des données.

La FIGURE 1.1 montre la différence entre un réseau distribué et décentralisé. Dans le réseau distribué, un nœud central gère la coordination entre les nœuds distribués. Dans le réseau décentralisé, chaque nœud peut demander ou envoyer des informations aux nœuds adjacents.

Technologie Blockchain

La technologie Blockchain réfère à une structure de données distribuées particulières, ainsi qu'à un protocole qui permet l'évolution de cette structure de données. Les Blockchains assurent donc une transmission d'informations à travers différents noeuds d'un réseau. Ce qui fait leur particularité est que ce fonctionnement ne repose pas sur un organe de contrôle particulier. En particulier, et

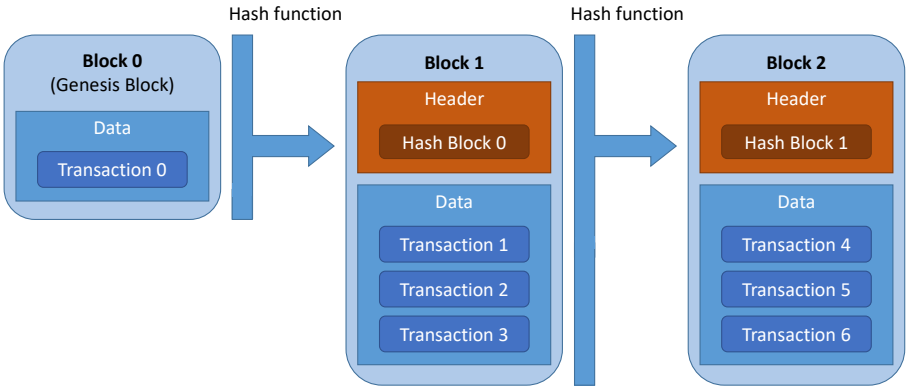

FIG. 1.2 : Représentation d'une Blockchain. Chaque bloc inclut le hash du bloc précédent dans ses entêtes.

contrairement aux systèmes classiques, les Blockchains sont construites de façon à ne pas avoir à faire confiance en les noeuds qui font fonctionner le système.

La structure de données est divisée en blocs, qui sont créés à intervalles réguliers et ajoutés à la Blockchain. Un bloc contient des entêtes, ainsi qu'une liste de transaction, qui représentent les changements atomiques de l'état de la Blockchain. Comme la FIGURE 1.2 le montre, chaque bloc est lié de manière cryptographique au suivant par son hash, après validation suite à un processus d'obtention d'un consensus. Ce processus permet de s'assurer qu'aucune information ne peut être falsifiée. En effet, si un utilisateur cherche à modifier un bloc déjà propagé dans le réseau, son hash changera. Comme cette valeur de hash se retrouve dans l'ensemble des blocs suivants, l'utilisateur doit satisfaire les règles du protocole d'obtention de consensus pour l'ensemble de ces blocs, ce qui demande des ressources importantes.

Les Blockchains permettent à une communauté d'utilisateurs d'enregistrer des transactions dans un registre public pour cette communauté, de sorte qu'aucune transaction ne peut être modifiée une fois publiée. Les nouveaux blocs validés sont répliqués dans toutes les copies du registre de cette communauté, et quelconque conflit est résolu automatiquement selon des règles établies. La technologie Blockchain ouvre de nombreuses perspectives d'utilisation dans les autres applications comme la création de monnaies numériques, appelées cryptomonnaies.

Cette technologie forme ainsi un écosystème décentralisé d'applications ayant des propriétés intéressantes : la transparence des transactions, l'auditabilité des applications par tout utilisateur, ou encore la résistance à la censure [22]. Les applications décentralisées qui intègrent la technologie Blockchain afin de profiter de ces propriétés sont souvent appelées des *Decentralized Applications*

(DApps) [23]. Cette terminologie, issue de l'industrie, est également présente dans des contextes académiques. Nous utiliserons donc cette terminologie dans la suite de ce manuscrit. Une limitation de cette terminologie est qu'elle ne permet pas la distinction entre ce type d'application des autres applications décentralisées, qui n'intègrent pas la technologie Blockchain, telles que celles que nous avons vues précédemment.

Une Blockchain regroupe ainsi une structure de données répartie à travers de nombreux nœuds ainsi qu'un protocole déterminant les règles d'écritures de cette structure de données.

Problème des généraux Byzantins Avant l'apparition de Bitcoin, certains travaux utilisaient déjà des structures de données liant une chaîne d'information par des hashs. Ce concept était par exemple utilisé afin d'assurer l'horodatage d'informations [24].

Cependant, les Blockchains apportent un nouvel élément qui permet de retirer totalement le besoin de confiance en un tiers. Il s'agit du principe amené par les méthodes de consensus, telles que la preuve de travail, ou *Proof of Work* (PoW) [25] ou la preuve d'enjeu, ou *Proof of Stake* (PoS) [26]. Ces méthodes permettent de résoudre un problème, appelé « Problème des généraux Byzantins » [27], relatif à l'intégrité des messages échangés entre plusieurs nœuds d'un réseau. Il s'agit donc d'une solution mise en place afin d'éviter les pannes byzantines. En général, les architectures distribuées se focalisent uniquement sur les erreurs de transmission ou les indisponibilités, temporaires ou permanentes, d'un ou plusieurs nœuds du réseau. Les Blockchains se focalisent également sur les effets de la présence de nœuds malicieux, qui peuvent volontairement altérer ou ne pas envoyer leurs messages.

Cependant, une faute byzantine n'est pas forcément causée par un acteur humain malicieux : il s'agit de tout comportement inattendu et contradictoire sur le réseau. Pour concevoir un système distribué tolérant aux pannes byzantines, il faut donc prévoir qu'un nœud peut se comporter différemment selon le nœud avec lequel il communique.

On dit alors qu'un système distribué est tolérant aux pannes s'il peut fonctionner avec un certain nombre de nœuds indisponibles. Il est de plus tolérant aux pannes byzantines s'il peut fonctionner avec un certain nombre de nœuds agissants de manière imprédictible.

La FIGURE 1.3 présente le cas d'un système décentralisé possédant des nœuds byzantins. Ces nœuds ne se comportent pas de façon honnête : ils envoient des informations factices, ou bien ne suivent pas le protocole du réseau correctement. Si le réseau considéré est tolérant aux pannes byzantines, alors le protocole protégera automatiquement les nœuds honnêtes du comportement des nœuds byzantins.

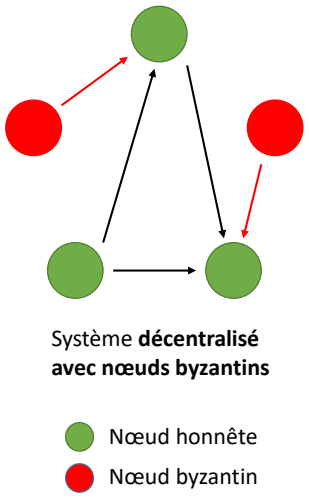

FIG. 1.3 : Représentation graphique d'un nœud byzantin au sein d'un réseau.

Méthodes de consensus Depuis 2008, de nombreux algorithmes ont été développés et analysés pour atteindre le consensus entre les différents nœuds du réseau sur les blocs valides composant la Blockchain. Ce consensus est en effet nécessaire au bon fonctionnement du réseau.

Le premier, la PoW [25], est la principale méthode de consensus utilisée, notamment par les Blockchains Bitcoin et Ethereum. Dans la PoW, chaque nœud du réseau peut choisir de devenir un mineur. Chaque mineur peut proposer un bloc à ajouter à la structure de données commune. Cependant, pour qu'un bloc soit valide, il doit posséder un hash particulier, qui valide une contrainte donnée par le protocole. Une contrainte classique est la nécessité pour ce hash, vu comme un simple nombre binaire, d'être inférieur à une valeur cible, appelée *difficulté*. Cette difficulté varie selon les règles définies par le protocole du réseau. Dans le réseau Bitcoin, la difficulté est ajustée afin d'obtenir un nouveau bloc toutes les 10 minutes en moyenne (d'après la page [1] au 09/09/2021). Cette durée, appelée « temps moyen de création d'un bloc », est actuellement de 14 secondes pour Ethereum (d'après la page [2] au 09/09/2021).

Le hash d'un bloc étant déterministe, il est nécessaire d'ajouter un nombre au sein du bloc, appelé *nonce*, que les mineurs peuvent faire varier afin d'influer sur le hash du bloc. Les mineurs étant rémunérés lorsqu'ils trouvent un bloc accepté par le réseau, ils sont en compétitions les uns avec les autres. L'objectif de la PoW est de rendre économiquement difficile de changer l'historique des blocs du réseau. Si un attaquant souhaite modifier un des blocs acceptés par le

[1] `https://data.bitcoinity.org/bitcoin/block_time`
[2] `https://ethstats.net`

réseau, il devra également rendre les blocs suivants valides, ce qui demande une puissance de calcul trop importante en pratique. En effet, l'ensemble des nœuds du réseau étant en compétition pour trouver des blocs valides, la probabilité que le même acteur trouve un grand nombre de blocs valides consécutivement, et sans que d'autres acteurs n'en trouvent, décroît de façon exponentielle avec le nombre de blocs. C'est donc l'algorithme de consensus qui permet de sécuriser le réseau, et ce de façon probabiliste : plus un bloc est ancien, moins il aura de chance d'être modifié ou supprimé de la Blockchain.

Ainsi, la PoW est un algorithme de consensus particulier. Un algorithme de consensus est un algorithme qui assure la résolution du problème des Généraux Byzantins. C'est-à-dire qu'il s'agit d'un algorithme qui permet d'assurer le consensus sur l'état de la Blockchain parmi l'ensemble des nœuds du réseau. La PoW n'est pas la seule méthode de consensus utilisée. D'autres algorithmes de consensus sont par exemple :

- La PoS [26]. Avec la PoS, les nœuds pouvant proposer les blocs ne sont plus choisis en fonction de la puissance de calcul qu'ils possèdent. À la place, si un utilisateur veut pouvoir proposer des blocs au reste du réseau, il doit bloquer un certain montant de cryptomonnaie, appelé « enjeu », qu'il pourra récupérer lorsqu'il ne souhaitera plus produire de blocs. Ainsi, plus un utilisateur à un enjeu élevé, plus il aura de chances de produire le prochain bloc. L'intérêt de cet enjeu est de pouvoir éliminer tout ou partie de cet enjeu si le réseau prouve que l'utilisateur agit de façon malicieuse. Un des avantages de la PoS sur la PoW est la réduction des besoins énergiques de l'algorithme [28]. En effet, la PoW nécessite pour fonctionner que le réseau utilise une quantité importante de ressources de calcul, sous la forme de cartes graphiques ou de *Application-Specific Integrated Circuit* (ASIC) qui calculeront un maximum de hashs pour le processus de minage des blocs. Cette consommation n'est pas présente lors de l'utilisation de la PoS.

- La preuve d'enjeu déléguée, ou *Delegated Proof of Stake* (DPoS) [29] fonctionne sur un principe similaire à la PoS, mais possède un nombre de producteurs de blocs fixe pour un temps donné. Les autres utilisateurs peuvent alors choisir de déléguer leur enjeu à un de ces producteurs de blocs. Cette approche est plus simple à mettre en place que la PoS, mais augmente la centralisation de la Blockchain [30].

- La preuve d'autorité, ou *Proof of Authority* (PoA) [31]. Cette forme de consensus est centralisée, puisque seul un nœud spécifique peut proposer des blocs au réseau. Il s'agit ainsi de la principale méthode de consensus utilisée par les Blockchains privées utilisées par certains groupes industriels, puisque l'aspect de la décentralisation est moins important pour ces cas d'applications.

- Ou encore des algorithmes de consensus tolérants aux fautes byzantines mais non issus des recherches sur les Blockchains, tels que Raft [32]. Cet

algorithme cherche à élire un représentant que les autres nœuds devront suivre.

Structures de données Comme énoncé précédemment, une Blockchain est un registre distribué de données. Afin de comprendre les propriétés de cette technologie et son intérêt, il peut être intéressant de l'abstraire en tant que structure de données, avec des propriétés bien précises :

- L'immuabilité. Une fois acceptée par le réseau, aucune donnée de la Blockchain ne peut être modifiée ou supprimée [33].
- La pseudonymité ou anonymité des utilisateurs dans une Blockchain publique. Les utilisateurs sont en effet distingués par une adresse dérivée de leur clé publique, et aucune entité n'est chargée de vérifier l'identité des participants [34].
- La traçabilité des transactions [35]. L'ensemble des opérations effectué sur la Blockchain peuvent être vérifiées et suivies.
- La fiabilité, ainsi que la non-nécessité de faire confiance en un tiers. En effet, tout utilisateur d'une Blockchain peut se connecter à ce réseau pair-à-pair par le biais d'un programme appelé client Blockchain. Ce client assure la synchronisation des informations avec les nœuds voisins, la validation des données reçues, et la signature des transactions réalisées par l'utilisateur. Si ce programme, auditable librement par tous, effectue correctement ces différentes tâches, un utilisateur à l'assurance de sa bonne participation au réseau. Ainsi, il ne risque par exemple pas d'être trompé par un acteur malicieux qui lui fournirait des informations falsifiées [36].

Cependant, ces propriétés ne sont vérifiées que si le mécanisme assurant le consensus n'échoue pas. Cela signifie qu'un nombre suffisant des nœuds du réseau suit les règles du protocole sous-jacent à la Blockchain.

En plus d'être une structure de données, une Blockchain abstrait également souvent ses propres données au sein des concepts suivants :

- Les transactions, qui sont des messages signés par un utilisateur, et transmis à l'ensemble du réseau.
- Les blocs, composés des transactions qui sont validées par le nœud du réseau choisi par la méthode de consensus.
- L'état, qui peut correspondre à l'ensemble des états des variables et des codes des contrats intelligents d'une Blockchain supportant cette fonctionnalité.
- Des structures de données basées sur les hashs des données, tels les arbres de Merkle [37], qui forment la structure sous-jacente à une Blockchain et permet de rapidement vérifier une transaction.

Cryptographie De nombreux travaux relatifs à la cryptographie sont utilisés au sein de la technologie Blockchain. Tout d'abord, la signature des transactions est réalisée avec des outils de cryptographie asymétrique. Par exemple, les signatures Ethereum et Bitcoin reposent sur l'*Elliptic Curve Digital Signature Algorithm*, ou ECDSA [38]. Un compte Ethereum possède ainsi une clé publique, faisant office d'adresse, et d'une clé privée utilisée pour signer les transactions de l'utilisateur.

Les fonctions de hashage sont également beaucoup utilisées au sein des Blockchains, puisqu'elles permettent de relier les différents blocs entre eux. L'objectif de ces fonctions est de produire rapidement une chaîne d'octet de taille constante à partir d'une certaine entrée. Toute modification de l'entrée, même minime, mènera à un résultat entièrement différent en sortie. De plus, le calcul de l'entrée à partir du résultat est actuellement impossible à réaliser en pratique pour les fonctions de hashage utilisées. Ethereum utilise par exemple Keccak256 [39] comme fonction de hashage qui relie deux blocs.

Contrats intelligents En plus des transferts décentralisés de valeur monétaire par les cryptomonnaies, les Blockchains peuvent aussi agir en tant que registres d'information, pouvant assurer par exemple la gestion d'identité et d'authentifications d'une personne. Une autre des fonctionnalités de certaines Blockchains est la possibilité d'exécuter des programmes sur la Blockchain. Ces programmes sont appelés « contrats intelligents », ou *smart contracts*, et leur exécution a les mêmes propriétés que les autres transactions : immuabilité, pseudonymité, traçabilité et fiabilité. Ce concept, initialement proposé en 1997 par Nick Szabo [40], peut profiter grandement des propriétés des Blockchains. Vitalik Buterin propose en 2014 Ethereum [2], la première Blockchain dédiée aux contrats intelligents. Hyperledger Fabric [41] est un autre exemple d'une telle Blockchain, poussée par *International Business Machines Corporation* (IBM) et le consortium Hyperledger de la fondation Linux. Cette dernière est une Blockchain privée, ce qui signifie que l'algorithme de consensus utilisé repose sur la confiance que les utilisateurs ont envers les administrateurs du réseau. Ainsi, l'exécution d'un contrat intelligent sur Hyperledger Fabric n'est pas décentralisée, contrairement à un contrat intelligent sur Ethereum. D'autres technologies appartenant à la catégorie des registres distribués, ou *Distributed Ledger Technologies* (DLT), dont les Blockchains font partie, peuvent proposer des contrats intelligents. El Ioini et Pahl [42] présentent les différents types de DLT qui existent à l'heure actuelle, telles que les Blockchains, *Tangle*, les *Hashgraph* ou encore les *Sidechain*.

Lorsqu'un utilisateur veut exécuter une des fonctions d'un contrat intelligent, par exemple sur Ethereum, il réalise une transaction classique, mais ajoute des données supplémentaires permettant aux nœuds validant les transactions d'exécuter la fonction choisie. Une fois la transaction diffusée, les mineurs vont exécuter le code du contrat et mettre à jour l'état des variables stockées sur la Blockchain. Les traces d'exécutions sont également stockées. L'intérêt des

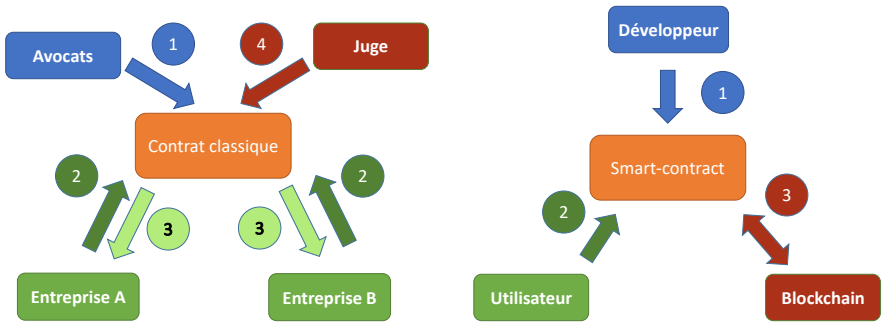

FIG. 1.4 : Comparaison entre le fonctionnement d'un contrat classique à gauche et celui des contrats intelligents à droite

contrats intelligents, en plus de ceux mentionnés précédemment, est d'intégrer l'ensemble des fonctionnalités nécessaires à la gestion des paiements au sein des environnements d'exécution des programmes. Ainsi, il est possible de programmer des contrats pour des paiements conditionnels, des échanges atomiques de valeur entre utilisateurs, etc. Il est également possible de créer des contrats intelligents qui permettent la création de *Decentralized Autonomous Organization* (DAO), comme le Dai [43], qui sont des entités fonctionnant comme des entreprises (avec des clients, fournisseurs et actionnaires), mais de façon autonome. Leur fonctionnement est alors régi entièrement par des contrats intelligents, sur lesquels des utilisateurs peuvent avoir un certain contrôle (comme le vote).

De nombreux concepts liés aux contrats intelligents, comme la vérification formelle [44], l'analyse statique [45], ou encore de nouvelles fonctionnalités telles que la parallélisation de l'exécution des contrats [46], sont en étude.

La FIGURE 1.4 montre la différence entre les concepts de contrats intelligents et les contrats classiques. Un contrat classique, par exemple entre deux entreprises, fonctionne comme suit :

1. Un ou plusieurs avocats rédigent le contrat.
2. Le contrat est alors signé par l'ensemble des parties.
3. Chaque partie exécute ensuite ses devoirs sur la base du contrat.
4. S'il y a un litige sur l'exécution du contrat, un juge peut faire exécuter le contrat ou imposer des dommages et intérêts.

Un contrat intelligent utilise un paradigme différent :

1. Un développeur écrit un smart-contrat et le déploie sur une Blockchain.
2. Un utilisateur peut appeler les différentes fonctions du contrat. Pour cela, il signe des transactions sur la Blockchain.

3. À chaque appel, la Blockchain se charge de la bonne exécution du contrat automatiquement.

Dans le cas d'un contrat intelligent, le programme informatique déployé sur la Blockchain fait foi du contrat entre l'utilisateur et le développeur. Le langage est informatique plutôt que juridique. Il ne peut alors pas y avoir en théorie de litige puisque ce code est exécuté dans le cadre du protocole de la Blockchain.

Cependant, des vulnérabilités présentes dans ce programme, volontairement ou non, peuvent toutefois tromper les utilisateurs. Dans ce cas, l'exécution du contrat intelligent sera correcte, mais ne sera pas forcément ce à quoi s'attendait l'utilisateur. Dans ce cas, une action en justice pourrait être considérée par l'utilisateur.

Globalement, nous pouvons dire que les contrats intelligents apportent de l'automatisation des processus, ainsi qu'un lien direct avec la gestion de valeur des biens transmis par le contrat.

Écosystème d'outils liés aux Blockchains Les Blockchains possèdent un écosystème d'outils permettant de faciliter l'interaction utilisateur. La méthode classique pour interagir avec une Blockchain est d'exécuter sur sa propre machine un client de cette Blockchain. Ce client est un noeud du réseau, et suis le protocole en communiquant avec le reste du réseau, en pair-à-pair. Cependant, certaines entreprises, telles qu'Infura[3], proposent l'utilisation tierce de leur propre nœud.

D'autres outils utiles pour la technologie Blockchain sont les explorateurs de blocs, tels que Etherscan[4]. Un explorateur de blocs permet de consulter un ensemble d'informations relatives à la Blockchain, mais surtout consulter toutes les transactions effectuées. Ainsi, un utilisateur peut confirmer qu'une transaction qu'il a réalisée a bien été acceptée par la Blockchain.

Enfin, d'autres outils permettent l'aide au développement de contrats intelligents. Ainsi, la suite Truffle[5] propose des environnements de développement, de tests, et de déploiement de smart contracts sur Ethereum.

1.2.2 Limitations de la technologie Blockchain

Le domaine de la Blockchain étant jeune et peu mature, il existe de nombreuses limitations à cette technologie. Nous allons aborder deux des principales limitations, la scalabilité et l'interopérabilité.

[3] `https://infura.io`
[4] `https://etherscan.io`
[5] `https://trufflesuite.com`

Scalabilité

Selon le Canadian Institute of Chartered Accountants, la scalabilité désigne l'aptitude d'un produit ou d'un système de conserver ses propriétés fonctionnelles malgré un changement de taille ou de certains paramètres [47]. Une des principales limitations des Blockchains publiques actuelles est le manque de scalabilité. Les capacités d'une Blockchain dépendent en effet des choix crypto économiques des développeurs, de la méthode de consensus choisie, et de nombreux autres paramètres. Le terme de *scalabilité* est souvent utilisé avec différentes significations selon le contexte et le cas d'application. Nous allons donc détailler les différents types de problèmes de scalabilité qui limitent la technologie Blockchain.

Latence et finalité Tout d'abord, le regroupement de transactions au sein de blocs introduit une latence non négligeable dans les échanges d'information, ce qui limite les applications possibles aux cas d'utilisation demandant des transferts en temps réel. Dans un contexte Blockchain, nous définissons la latence comme étant le délai entre le moment où un utilisateur émet une transaction, et le moment où les autres utilisateurs considèrent cette transaction comme faisant partie de la Blockchain. Cette latence dépend de la fréquence des blocs dans la Blockchain considérée, mais également du nombre de confirmations nécessaire pour assurer la sécurité du réseau. Comme nous l'avons vu dans le paragraphe 1.2.1 Méthodes de consensus, les temps moyens de création d'un bloc respectifs de Bitcoin et d'Ethereum sont de l'ordre de 10 minutes et 14 secondes, ce qui donne des latences très élevées. Nous définissons alors la sécurité d'une Blockchain comme étant sa capacité à assurer l'immuabilité des transactions faites sur le réseau.

Dans ce contexte, le nombre de confirmations correspond au nombre de blocs qui ont été créés à la suite du bloc considéré. Comme nous l'avons vu en paragraphe 1.2.1 Méthodes de consensus, la sécurité d'une Blockchain est souvent probabiliste, et les informations contenues dans un bloc ancien seront considérées comme *plus sûres* que celles d'un bloc plus récent. Cependant, certains modèles de consensus assurent la *finalité* des transactions [48]. Cela signifie qu'après un certain temps fixé, la sécurité des données est assurée.

Nombre de transactions et coûts Une autre limitation est le nombre brut de transactions que le réseau peut traiter sur un temps donné, qui correspond au nombre de transactions que l'on peut insérer dans un bloc multiplié par la période de création des blocs.

Ainsi, si trop d'utilisateurs souhaitent réaliser simultanément des transactions par rapport aux capacités du réseau, le coût des frais de transactions augmente, ce qui rend impossibles certaines applications potentielles demandant un grand nombre d'échanges. Par exemple, la Blockchain Bitcoin possède une limite théorique de 7 transactions par secondes pour l'ensemble du réseau [49].

Complexité des calculs et stockage Enfin, la complexité des calculs réalisables lors d'une transaction est dépendante des choix d'implémentation de la Blockchain. Elle dépend en particulier du type de méthode de consensus et des instructions-machine supportées. Par exemple, Bitcoin peut difficilement gérer des transactions complexes. Das et al. [50] mentionne en effet : « une limitation particulièrement importante de Bitcoin est son support limité pour les smart contracts ».

En ce qui concerne le stockage de données, nous pouvons donner l'exemple d'Ethereum, la principale Blockchain pour l'utilisation de contrats intelligents, mais dont le stockage de données coûte très cher, comme le mentionnent Gupta et al. [51]

Interopérabilité dans un contexte Blockchain Afin de résoudre les problèmes décrits précédemment, de nombreux projets Blockchains ont vu le jour depuis l'apparition de Bitcoin. Le site CoinMarketCap [52] répertorie en effet 1058 Blockchains différentes au 7 septembre 2021. Le domaine étant peu mature, ces projets utilisent pour la majorité des protocoles et des écosystèmes d'outils différents. Cela mène à une autre limitation importante de la technologie et de son écosystème et le manque d'interopérabilité, et ce à plusieurs niveaux. Nous discutons de cette limitation au sein de la section 1.4.3, après avoir défini les concepts de base de l'interopérabilité.

1.3 Concepts liés à l'interopérabilité

Wegner [53] définit l'interopérabilité comme étant la capacité pour deux composants logiciels ou plus de coopérer en dépit des différences de langages, d'interface, et de plateforme d'exécution. Ce concept peut être décliné sur différents niveaux que nous détaillons à travers les sous-sections suivantes.

Tolk et Muguira [54] proposent en effet un modèle, appelé *Levels of Conceptual Interoperability Model* (LCIM), des différents niveaux d'interopérabilité conceptuelle qu'ils ont identifiés. Nous représentons les différents niveaux d'interopérabilité de ce modèle FIGURE 1.5.

De nombreux autres modèles d'interopérabilité des systèmes ont été élaborés. Par exemple, Chiu [55] détaille un exemple d'utilisation du modèle *Levels of Information Systems Interoperability* (LISI). Ce modèle est plus complexe que LCIM. En effet, il cherche à juger l'interopérabilité d'un système d'information selon 5 niveaux, dont chacun affecte 4 systèmes différents : les processus, les applications, les infrastructures et les données. Nous ne considérons pas ce modèle dans la suite de ce mémoire, mais il est important à considérer dans les cas de systèmes d'information complexes, pour lesquels les différentes parties du système peuvent avoir des considérations d'interopérabilité très différentes.

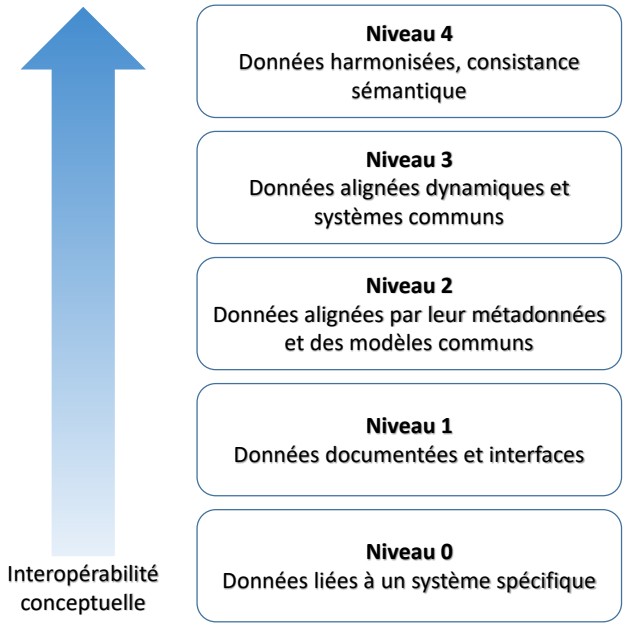

FIG. 1.5 : Schéma descriptif de différents niveaux d'interopérabilité, adapté de [54]

Finalement, Figay [56] conclut que chacun de ces modèles d'interopérabilité existants a des particularités sur la façon dont ils modélisent les systèmes d'information. Il précise également que ces modèles sont plutôt génériques, et qu'il est nécessaire d'aller en profondeur pour correctement étudier l'interopérabilité d'un système.

Il faut ainsi considérer le plus pertinent de ces modèles pour l'étude de l'interopérabilité d'un système donné. Nous partons ainsi initialement du modèle LCIM. Par mesure de simplicité, nous considérons dans la suite trois niveaux d'interopérabilité. Tout d'abord, l'interopérabilité des données, dont l'objectif est de proposer des interfaces permettant à un système de communiquer avec d'autres. Il s'agit du niveau 1 du LCIM. Ensuite, nous considérons l'interopérabilité syntaxique, visant à unifier les interfaces de différents systèmes via des modèles communs, plus ou moins complexes et puissants. Il s'agit des niveaux 2 et 3 du LCIM. Enfin, l'interopérabilité sémantique cherche à proposer un cadre dans lequel la sémantique des données, c'est-à-dire leur sens et leurs relations sont unifiés. Il s'agit du 4ème niveau décrit par le LCIM.

Approche des données

L'approche des données, en interopérabilité, correspond à l'élaboration d'interfaces et d'*Application Programming Interface* (API) permettant des partages de données directs entre des systèmes. Si les systèmes considérés n'utilisent pas les données voulues de la même façon, il faut alors des modules de traduction de données.

Un exemple d'une telle interopérabilité se retrouve dans les implémentations de services web *REpresentational State Transfer* (REST), popularisé par Richardson et Ruby [57]. Un site internet peut facilement proposer une interface aux données et fonctionnalité du site via une API REST. Par exemple, il est possible de poster automatiquement des nouveaux *tweets* sur Twitter via leur API [58]. Cela peut être utilisé par un créateur de blog pour *tweeter* automatiquement lors de la publication de nouvelles informations.

Une des principales limitations de cette approche est la nécessité de rendre interopérables les systèmes deux à deux. Si trois systèmes A, B et C, d'API respectives API_A, API_B et API_C, veulent communiquer, alors il faut créer les interfaces entre API_A et API_B, entre API_A et API_C ainsi qu'entre API_B et API_C. La difficulté est alors exponentielle en fonction du nombre de systèmes à rendre interopérable.

Approche syntaxique

L'approche syntaxique, quant à elle, vise à élaborer des modèles permettant d'inférer directement des moyens de communication adaptés entre plusieurs systèmes. Ces modèles doivent alors être assez complets afin de pouvoir éviter des pertes d'information lorsque les deux systèmes communicants n'ont pas le même point de vue sur une donnée. Un exemple simple est l'envoi de messages électroniques (email). Pour que deux personnes puissent s'échanger des emails, il faut qu'elles utilisent des clients supportant le même standard, comme *Simple Mail Transfer Protocol* (SMTP). Dans ce cas, deux systèmes peuvent communiquer correctement seulement s'ils implémentent le même standard. Ce standard est assez générique pour être utilisé par l'ensemble des clients mail existants.

De plus, il est possible d'élaborer les modèles de plusieurs manières. L'une d'entre elles est de créer un modèle unique à une donnée, qui regroupe l'ensemble des informations relatives à cette donnée. Ensuite, il faut que chaque système traduise ce modèle unique en un modèle adapté à son point de vue.

Une autre façon de procéder est de modéliser l'information séparément pour chaque système. Ensuite, il faut alors créer des règles de cohérence afin de regrouper ces différents modèles en une notion commune.

Approche sémantique

Enfin, l'approche sémantique, plus complète, ne vise pas seulement à réaliser des échanges d'information entre les systèmes. L'objectif est d'assurer également que le *sens* des informations transmises soit conservé. Pour cela, il faut abstraire les notions relatives aux données à échanger à travers une *ontologie*. Une ontologie cherche de manière générale à modéliser les relations entre toutes les notions relatives à un domaine particulier, mais il est également possible de créer une ontologie qui couvre plusieurs domaines différents.

Une ontologie relative à un domaine devrait également permettre d'être réutilisé dans d'autres cas d'usage, comme a cherché à faire Simperl [59] dans le cas du web sémantique.

Pour le développement des ontologies, un langage couramment utilisé est *Ontology Web Language* (OWL) [60]. Ce langage est décliné en trois versions : OWL Full, OWL Light et OWL *Description Logics* (DL). Ces trois versions ont des fonctionnalités et propriétés différentes. Par exemple, OWL DL est moins expressif que OWL Full, mais en contrepartie il assure la calculabilité de toute expression, ce qui n'est pas le cas pour OWL Full.

L'objectif du langage OWL est de permettre la formalisation de concepts et de leurs relations. Il est alors possible d'utiliser des raisonneurs [61] afin de déduire de nouveaux liens entre les concepts définis.

Conséquences de l'interopérabilité sur les systèmes informatiques

Les motivations pour améliorer l'interopérabilité des systèmes sont nombreuses. Dans un premier temps, l'interopérabilité entre services permet une meilleure coopération entre ces services. Il devient plus facile de réutiliser des données existantes, et, comme le précise Figay [56], cela optimise alors l'utilisation des ressources de chaque service.

Un autre avantage de systèmes interopérable est la cohérence des données. Par exemple, dans le domaine médical, Iroju et al. [62] précisent l'intérêt du partage plus efficace des données d'un patient. Cela permet d'améliorer son suivi, en évitant que ses données soient fragmentées. Un dossier médical partagé diminue également le risque de présence d'erreurs humaines lors de diagnostics effectués à partir de données incomplètes.

L'interopérabilité sémantique apporte également de nouvelles possibilités. Tout d'abord, elle permet de s'assurer que les données partagées entre différents systèmes représentent bien les mêmes concepts. Le cas échéant, une ontologie permet ainsi de détecter et corriger des utilisations incohérentes de concepts proches. Comme le montre Meilicke [63], de nombreuses techniques permettent de détecter les incohérences lors de la mise en correspondance de deux ontologies.

Selon Livet et al. [64], une ontologie dans le domaine des modèles à base d'agents permet par exemple d'améliorer les processus de développement et de créations de modèles. De plus, cela permet de décrire formellement les modèles créés, ce qui rend plus rigoureuse la présentation des hypothèses et améliore ainsi la reproductibilité des modèles.

Finalement, l'ensemble des motivations présentées justifie la recherche d'une meilleure interopérabilité des systèmes.

1.4 L'interopérabilité des systèmes Blockchains

Après avoir défini précédemment certains concepts liés à l'interopérabilité et aux systèmes Blockchains, nous allons à présent étudier l'application de l'interopérabilité à ces derniers.

1.4.1 Motivations

Comme mentionné dans la section précédente, l'interopérabilité joue un rôle important au sein des systèmes d'information. Nous pouvons donc penser que ceci est également vrai dans le cas des systèmes Blockchains. Les Blockchains permettent une décentralisation et une sécurisation du partage de données d'une part, mais du partage de valeur d'autre part. Ainsi, la question de l'interopérabilité au sein de ces systèmes est multiple.

Tout d'abord, l'interopérabilité des données entre plusieurs Blockchains permettrait une meilleure cohérence des données au sein des Blockchains. Par exemple, pour le cas d'application de la sécurisation de diplômes sur la Blockchain [65], une université peut certifier des informations relatives aux diplômes qu'elle délivre. Ces diplômes pourraient alors être indifféremment stockés et utilisés sur une Blockchain ou sur une autre.

Mais c'est le partage de valeur qui est un point critique de l'interopérabilité des Blockchains. En effet, le partage de valeur à travers différentes Blockchains permettrait de nombreuses possibilités :

- Simplifier l'expérience des utilisateurs, puisque ceux-ci pourraient échanger avec l'ensemble des utilisateurs de Blockchain via une unique interface et un unique compte Blockchain.
- Augmenter la scalabilité des Blockchains en proposant le *sharding* des transactions [66]. Le *sharding* est une technique visant à paralléliser les transactions d'une Blockchain sur différents *shards*. Chaque *shard* ne traite alors qu'une partie des transactions du réseau, et les communications entre *shards* permettent une augmentation importante des capacités totales du réseau.

- Proposer au sein d'une Blockchain les fonctionnalités d'une autre. Par exemple, grâce à BTC Relay [67], les utilisateurs de contrats intelligents sur Ethereum peuvent payer avec des Bitcoins, alors que la Blockchain Bitcoin ne possède pas de contrats intelligents évolués.

Enfin, l'interopérabilité sémantique parfaite au sein de l'écosystème Blockchain permettrait pour une certaine application d'utiliser plusieurs Blockchains ou services Blockchain, avec des propriétés et caractéristiques différentes, et ce de façon transparente pour l'utilisateur. En effet, si l'on définit formellement les concepts de transaction pour deux Blockchains différentes, alors l'application peut réaliser des transactions sur l'une ou l'autre des deux Blockchains indifféremment. Le développement de telles applications est alors rendu plus facile. Belchior [68] mentionne ainsi que le développement d'une solution d'interopérabilité entre les Blockchains appelée « Blockchain of Blockchains » permet de se rapprocher d'une utilisation de la technologie Blockchain agnostique des Blockchains utilisées.

1.4.2 Impact des caractéristiques des systèmes Blockchain sur leur interopérabilité

Après avoir étayé les motivations de l'interopérabilité au sein de l'écosystème des Blockchains, nous pouvons étudier si leurs caractéristiques techniques ont une influence sur leur interopérabilité.

Nous avons vu que les Blockchains sont des systèmes décentralisés. Cela implique qu'un grand nombre d'acteurs doivent coopérer afin d'établir le consensus voulu sur l'état d'une Blockchain. Cela signifie qu'une solution d'interopérabilité entre deux Blockchains doit prendre en compte cet aspect, qui revient ainsi à une interopérabilité entre de nombreux systèmes.

De plus, la pseudonymité ou anonymité des utilisateurs dans une Blockchain publique à pour conséquence que l'interopérabilité entre des Blockchains doit permettre l'authentification des utilisateurs à partir d'une unique clé publique.

Par ailleurs, les Blockchains fonctionnent en environnement fermé. Cependant, nous avons précisé au sein de notre article [69] qu'elles doivent parfois interagir avec d'autres services liés, tels que des systèmes de stockage de données, distribuées ou non, ou des solutions de scalabilité des Blockchains. Un des problèmes à considérer est que l'écosystème lié est très jeune, et de nouveaux services apparaissent régulièrement. L'interopérabilité doit ainsi être évolutive, afin de prendre en compte de nouveaux services qui sont régulièrement lancés.

Enfin, une caractéristique fondamentale des Blockchains est la non-nécessité de faire confiance en un tiers. Il faut ainsi éviter au maximum qu'une solution d'interopérabilité des Blockchains réintroduise ce besoin pour son fonctionnement.

1.4.3 Limitations actuelles de l'interopérabilité dans le contexte Blockchain

Comme mentionné au sein de la section 1.2.2, la Blockchain et son écosystème ont aujourd'hui une interopérabilité limitée. De nombreux projets coexistent dans le domaine, mais beaucoup reposent sur des technologies et des interfaces différentes. Nous allons voir que cela provoque des limitations en termes d'interopérabilité sur plusieurs niveaux : entre les différentes Blockchains, entre les projets au sein d'une même Blockchain, ainsi qu'entre les composants d'une application donnée intégrant la technologie Blockchain.

Une Blockchain se distingue alors des autres par des différences de cas d'utilisation, de fonctionnalité, de vision, de niveau de maturité, de méthode de consensus, ou encore de compromis sur les paramètres régissant le protocole, tels que les temps moyens de création d'un bloc ou la taille des blocs. Il en résulte que l'interopérabilité sémantique entre toutes ces Blockchains est complexe. Il en est de même pour de simples échanges de valeur entre deux cryptomonnaies. Belchior et al. [70] fournissent une revue précise de l'état de l'interopérabilité au sein des Blockchains en 2020. Ils donnent ainsi l'exemple des difficultés rencontrées pour rendre interopérables les Blockchains dites publiques et celles dites privées, fonctionnant avec des méthodes de consensus très différentes.

Aussi, de nombreux projets Blockchain ne cherchent pas à créer une nouvelle Blockchain, avec ses propres caractéristiques et fonctionnalités, mais cherchent à utiliser une Blockchain existante. Ainsi, ces projets n'ont pas à effectuer de recherches afin d'implémenter une Blockchain de toutes pièces. Cela mène ainsi à des problématiques de standardisation entre les différents projets fonctionnant sur une Blockchain spécifique.

Par exemple, la Blockchain Ethereum permet le développement de smart contracts. Ainsi, de nombreux projets Blockchain consistent à créer des smart contracts sur Ethereum, avec une interface utilisateur permettant d'interagir avec eux. Par conséquent, les développeurs d'Ethereum doivent faire face à des problèmes de standardisation, afin que tous ces projets puissent communiquer correctement entre eux, et que les outils créés pour cette plateforme soient réutilisables pour les différents projets. Par exemple, le site [6] recense au 14/09/2021 221 projets différents de finance décentralisée qui utilisent la Blockchain Ethereum. La gestion d'assets financiers entre l'ensemble de ces projets peut ainsi être rendue très complexe. Finalement, les Blockchains existantes ont également des niveaux de maturité différents, ce qui se traduit par une standardisation plus ou moins poussée au sein de leur écosystème.

Enfin, un dernier problème d'interopérabilité dans le contexte Blockchain est la communication entre une Blockchain et les autres systèmes nécessaires afin de concevoir une application décentralisée. En effet, un développeur souhaitant

[6] `https://defiprime.com/ethereum`

créer une application décentralisée peut intégrer la technologie Blockchain afin de tirer parti des propriétés des Blockchains : immuabilité, traçabilité, sécurité, etc. Cependant, il est également nécessaire d'intégrer d'autres composants, puisque les Blockchains ne sont pas adaptées à certaines tâches telles que le stockage de données volumineuses [71]. Des exemples de certains composants usuels d'applications décentralisées sont des solutions de stockage de données distribué, des bases de données distribuées, des librairies de développement usuelles ou encore des interfaces utilisateur adaptées. Aujourd'hui, la communication entre ces différents composants est généralement problématique. Par exemple, il n'existe pas à l'heure actuelle de standard permettant d'unifier les communications entre des implémentations différentes de tels services Blockchains [69].

Les interfaces existantes actuellement présentent également des limitations sémantiques : certaines caractéristiques des Blockchains, comme la finalité, ne sont pas prises en compte dans les autres composants, ce qui complexifie la tâche des développeurs qui doivent prendre en compte manuellement ces propriétés dans leur application. Dans ce cas d'interopérabilité, nous devons ainsi faire face à des problèmes d'architectures logicielles, de communication et partages de données, mais également de sémantique. Belchior et al. [70] mentionnent qu'il faudrait prendre en compte plusieurs finalités différentes pour s'adapter à la plateforme cible. Cela permet d'imiter, ou plutôt de contourner, l'interopérabilité sémantique entre deux Blockchains qui ont une finalité différente.

Ces différents niveaux d'interopérabilité dans un contexte Blockchain sont traités de façon inégale dans la littérature actuelle. En effet, comme nous allons le voir dans le chapitre suivant, la majorité des recherches se focalisent sur l'interopérabilité entre les Blockchains, et relativement peu de chercheurs travaillent sur l'interopérabilité au sein des applications décentralisées.

1.5 Contexte du domaine d'application de ces travaux

Cette section vise à présenter les liens entre les domaines de la Blockchain et du jeu vidéo. Nous motivons le domaine d'application choisi des recherches et précisons les conséquences que cela a sur l'interopérabilité des applications Blockchain décentralisées liées à ce domaine.

1.5.1 Caractéristiques du domaine d'application des jeux vidéo

L'industrie du jeu vidéo peut être caractérisée par la diversité des types de contenus proposés. En effet, les jeux peuvent être catégorisés selon le nombre de joueurs (un seul joueur, multijoueur, ou massivement multijoueur), le rythme du jeu (en temps réel ou tour par tour), les plateformes visées (mobile, console,

web ou PC), les modèles économiques de vente (vente du jeu, vente d'objets ou *skins* virtuels, revenus publicitaires), les mécaniques de jeu, etc.

Dans un jeu vidéo multijoueur classique, les joueurs se connectent sur un serveur centralisé. Même si certains éléments peuvent être partagés en pair-à-pair, l'architecture client / serveur est en effet privilégiée dans l'industrie [72]. Dans un premier temps, la majorité des données est téléchargée avec le jeu, en local, depuis les serveurs du jeu. La majorité des calculs, par exemple en termes de physique et rendu, sont effectués côté utilisateur. Les échanges de données entre le client et le serveur au cours du jeu consistent généralement en des messages de *chat* entre joueurs, des actions haut niveau des joueurs, comme des déplacements ou des tirs. Les calculs les plus critiques, comme les décisions de victoire d'une partie, sont effectués directement sur le serveur.

Une autre caractéristique de l'industrie du jeu est l'utilisation, de plus en plus commune, de moteurs de jeux afin d'accélérer le développement et de réduire son coût. Ils permettent en effet aux développeurs de ne pas a avoir à ré implémenter un moteur de rendu, un moteur de collision, ou encore un moteur physique. Comme le montrent Christopoulou et Xinogalos dans leur comparatif de moteurs de jeu [73], les moteurs les plus utilisés sont Unreal Engine [74] et Unity [75].

L'objectif étant de travailler sur l'interopérabilité entre les Blockchains et les autres composants nécessaires à la création d'applications Blockchain décentralisées, nous n'allons dans un premier temps pas chercher une implémentation générique des Blockchains dans le domaine du jeu. L'application du travail de thèse se fait avec les contraintes détaillées ci-après. Le jeu visé est multijoueur, avec un nombre de joueurs limité afin d'éviter des problèmes de scalabilité du nombre de transactions nécessaires. Il a pour objectif d'être temps réel, et donc des tests sont nécessaires afin de connaître la latence acceptable, probablement inférieure à la seconde. Enfin, les joueurs peuvent créer et acheter des *skins* graphiques de façon entièrement décentralisée.

Finalement, le besoin d'obtenir un grand nombre d'échanges de données volumineuses avec une latence minimale semble complexe à intégrer au sein d'une architecture distribuée utilisant la technologie Blockchain. Nous prenons ainsi cette contrainte en compte lors de l'implémentation souhaitée, en se limitant dans un premier temps à un scénario simple d'un jeu vidéo moderne multijoueur. Les perspectives de recherches présenteront alors les possibilités d'extrapolation des résultats obtenus à des jeux plus complexes à traiter dans notre contexte, tels que les jeux massivement multijoueurs.

1.5.2 Implication des caractéristiques du domaine dans le rôle de l'interopérabilité

Tout d'abord, une représentation sémantique des concepts liés à l'industrie du jeu vidéo est nécessaire, en parallèle de ceux liés à la Blockchain. Cela

permet d'analyser les interactions nécessaires entre les deux domaines, et ainsi étudier les points possibles d'amélioration de l'interopérabilité entre les composants requis à l'élaboration d'un jeu vidéo Blockchain. Pour cela, nous partons d'ontologies existantes des deux domaines. Pour la Blockchain, il s'agit de BLONDiE [76], EthOn [77], ainsi que Sandra [78] et sa surcouche CrystalSparkCannon [79]. Pour les jeux vidéo, il s'agit de Video Game Ontology [80] et Game Ontology Project [81].

Nous avons remarqué que les abstractions existantes ne peuvent pas décrire intégralement notre cas d'utilisation. En effet, les deux ontologies relatives au jeu vidéo ne prennent pas en compte l'écosystème complet du développement d'un jeu. Il est par exemple nécessaire d'y inclure les moteurs de jeux afin d'obtenir l'abstraction cohérente de l'architecture d'un jeu vidéo Blockchain.

De même, pour les ontologies relatives aux Blockchains, nous verrons dans le prochain chapitre que trop peu de notions sont explicitées et définies. Il manque ainsi les définitions de concepts tels que les modèles de consensus, les contrats intelligents, ou le stockage de données.

Cependant, ces ontologies peuvent fournir une base qu'il serait intéressant de compléter, autant la sémantique dans le domaine de la Blockchain, que de celle du jeu vidéo, ainsi que des interfaces nécessaires entre les deux domaines.

Chapitre 2

État de l'art

Ce chapitre présente l'état de l'art de l'interopérabilité des systèmes Blockchains, et plus particulièrement de l'interopérabilité des applications Blockchain décentralisées.

2.1 Formalisation des processus Blockchain

La formalisation des processus Blockchain est importante pour bien comprendre leur fonctionnement et aider au développement des applications qui intègrent cette technologie. Udokwu et al. [82] présentent une analyse de différentes méthodes utilisées pour le développement de DApp. Ils partent du constat suivant : les méthodes utilisées pour le développement logiciel ne sont pas toutes adaptées aux processus décentralisés. Ils retiennent également que très peu de développeurs utilisent des outils de modélisation lors de la conception de leur DApp.

En comparant différentes méthodes, ils montrent les avantages de certains outils de modélisation pour les environnements décentralisés. Tout d'abord, une modélisation orientée agent décrit les acteurs de l'application et leurs objectifs. Cela traduit ainsi une partie des besoins fonctionnels de la DApp. Ensuite, des modélisations à base d'*Unified Modeling Language* (UML) permettent de formaliser les interfaces des composants de l'application, ainsi que des diagrammes séquentiels des actions effectuées.

Des travaux similaires sont présentés par Jurgelaitis et al. [83]. En effet, ils utilisent les principes des architectures dirigées par les modèles pour formaliser les applications Blockchains. Cependant, leur méthodologie de modélisation consiste à utiliser des profils UML adaptés aux Blockchains.

Gull et al. [84] propose un *framework*, appelé Blockchain Oriented MOdel (BOMO), de développement de DApps orienté modèles. Ces travaux reposent

sur des outils graphiques qu'ils ont développés pour améliorer le développement de DApps. Leur proposition semble intéressante, mais ne repose pas sur des outils largement utilisés.

Une dernière approche de la littérature sur le sujet est proposée par Mavridou et Laszka [85]. L'objectif de leurs travaux est de développer des smart contracts en les représentant en tant que machine à état fini. Cette approche, plus technique, permet selon eux de développer de façon plus simple, automatisée et sécurisée les smart contracts. Le développement de smart contracts est une partie fondamentale du développement d'une DApp. Cependant, leur approche n'établit pas les autres étapes de développement, ou les interactions avec les autres services de l'application.

Dans la suite de ce manuscrit, nous considérons que la formalisation du fonctionnement d'une DApp sous forme de processus est la plus pertinente. En effet, ce type de modélisation est simple, mais permet d'étudier l'ensemble des interactions au sein d'une DApp. Nous choisissons ainsi d'utiliser la notation BPMN, décrite plus en détail dans la section suivante.

2.1.1 *Business Process Model and Notation* (BPMN)

Afin d'assurer une bonne communication entre les composants d'une application décentralisée intégrant une Blockchain, il est donc important de modéliser l'application, ses composants internes, et ses communications externes.

Une approche possible est de voir l'ensemble de ces éléments en tant que processus métiers. Les différentes modélisations existantes sont regroupées au sein de l'approche *Business Process Management*, ou BPM. Son objectif est d'obtenir une meilleure gestion des systèmes informatiques. Cela passe par exemple par une étude des flux de travail d'un système.

Le principal standard utilisé pour la modélisation des processus métier est BPMN [86], pour *Business Process Model and Notation*. Ce standard définit une représentation graphique permettant de modéliser des événements, des tâches, des décisions, ainsi que les gestions séquentielles ou parallèles d'un processus. Depuis sa version 2.0 [87], BPMN intègre également une sémantique d'exécution, permettant de suivre l'évolution de l'état d'un processus modélisé en fonction des événements et décisions prises. De nombreuses implémentations d'outils de modélisation et de moteurs BPMN existent. Nous pouvons par exemple citer Camunda[1] et Bonita[2], proposant tous deux des versions gratuites pour des projets open source ou de recherche.

Un exemple de modélisation par BPMN est présenté en FIGURE 2.1. Les cadres orange montrent les actions du Joueur 1, et les cadres bleus celles du

[1] `https://camunda.com`
[2] `https://fr.bonitasoft.com`

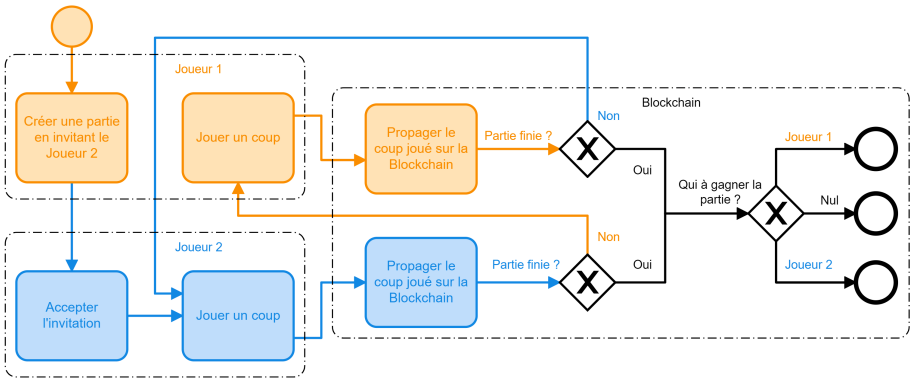

FIG. 2.1 : Exemple du BPMN d'un jeu Blockchain à deux joueurs en tour par tour.

Joueur 2. Les flèches montrent l'enchaînement des actions. Les losanges possédants une croix sont des bifurcations. Ainsi, après la propagation dans la Blockchain de chaque coup joué, nous devons évaluer les conditions de fin de jeu. Si elles ne sont pas complétées, alors on donne la main au joueur suivant. Si elles sont complétées, alors on détermine qui gagne, et on termine la partie.

Dans le contexte des applications Blockchains, BPMN peut alors être utile pour différentes tâches :

- Modéliser les interactions entre les composants d'une application Blockchain en fonction du cycle de vie de l'application.
- À un plus haut niveau, modéliser les interactions entre une application décentralisée spécifique, et les autres applications liées au cas d'usage en question. Par exemple, dans le cas d'une application dans le domaine du jeu vidéo, il faut étudier les interactions entre l'application décentralisée et les moteurs de jeu, utilisés pour le développement de la majorité des jeux vidéo.
- Suivre l'exécution d'une implémentation donnée sous des conditions spécifiques, afin de par exemple vérifier le maintient des propriétés des Blockchains à travers l'ensemble des composants de l'application.

En revanche, BPMN se basant sur des processus d'entreprise généraux, il n'est pas adapté à la modélisation de processus décentralisés. Il existe cependant des extensions à BPMN permettant de modéliser des processus décentralisés [88]. Néanmoins, le nombre d'instances de nœuds distinctes que l'on peut modéliser reste restreint, et inadapté à la modélisation des milliers de nœuds qui peuvent composer le réseau d'une Blockchain (par exemple, les plus de 8000 nœuds de la Blockchain Ethereum au 18/09/2019 selon la page [3]). Ainsi, un BPMN de-

[3] `https://www.ethernodes.org`

vra modéliser les différents types de nœuds d'un réseau, mais ne pourra pas modéliser l'ensemble des instances de ces nœuds.

Une particularité des systèmes décentralisés est le besoin d'interactions à deux niveaux : entre les utilisateurs et l'application d'une part, et entre les différents composants de l'application d'autre part. De plus, ces différentes interactions sont liées entre elles. Pour le comprendre, il faut analyser la manière dont chaque action réalisable par un utilisateur se propage dans l'application, et aux autres utilisateurs.

La formalisation d'une DApp via des modèles BPMN permet justement de comprendre comment les différents processus de l'application fonctionnent. Une modélisation complète de l'ensemble des interactions d'une application permet alors de savoir quelles interactions présentent des contraintes importantes lors de la conception de l'application.

2.1.2 Simulation de Systèmes Blockchains

En dehors de la formalisation de systèmes Blockchains par BPMN, une autre approche peut être considérée pour comprendre les interactions au sein d'une DApp. En effet, certains chercheurs décident, à la place de formaliser ces systèmes avec des modèles tels que BPMN, de simuler les différents acteurs liés à un système. C'est par exemple le principe de SimBlock [89]. Cet outil est basé sur des événements, et permet de simuler toutes les fonctionnalités associées à une Blockchain. Il est alors possible de paramétrer en profondeur la simulation, en faisant évoluer entre autres le nombre de nœuds de la Blockchain, les capacités de ces nœuds, les latences réseau entre ceux-ci, etc. SimBlock propose également un outil de visualisation de la propagation des informations dans le réseau.

Cependant, nous considérons que la formalisation par BPMN reste la plus générique. Entre deux applications ou systèmes Blockchains, de nombreux tests de paramètres doivent en effet être effectués pour s'adapter au cas considéré.

2.2 Les niveaux d'Interopérabilité des Systèmes Blockchains

Cette section vise à présenter l'interopérabilité dans un contexte Blockchain. Nous avons précisé précédemment les différents niveaux d'interopérabilité dans ce contexte : entre les différentes Blockchains, entre les applications et les services au sein d'une même Blockchain, ainsi qu'entre les composants d'une application donnée intégrant la technologie Blockchain.

2.2.1 Interopérabilité entre les Blockchains

Étant donné le nombre de Blockchains différentes aujourd'hui, l'interopérabilité entre ces dernières est primordiale au bon développement de l'écosystème. En effet, le paradigme des Blockchains prône le partage décentralisé de l'information et de la valeur. Cependant, si deux utilisateurs n'ont pas leur valeur sur la même Blockchain, il est difficile pour eux de s'échanger des informations ou de la valeur.

Ce manque d'interopérabilité participe à la création d'effet de réseau, qui entraîne une forte concentration des utilisateurs sur les Blockchains principales, comme le Bitcoin, simplement parce que de nombreuses personnes font déjà partie du réseau.

Depuis quelques années, de nombreuses recherches ont été réalisées afin de résoudre ce problème d'interopérabilité entre les différentes Blockchains existantes. Ces différentes recherches sont comparées dans Kannengießer et al. [90], dont le travail cherche à évaluer les différentes méthodes existantes afin de les recommander pour des cas d'usage spécifiques. Pour cela, ils commencent par identifier les méthodes utilisées pour gérer l'interopérabilité entre les Blockchains. Ensuite, ils définissent un ensemble de critères de comparaisons selon cinq axes : administration, flexibilité, réseau, performance et sécurité. Ils finissent par une comparaison entre les différentes approches d'interopérabilité selon ces axes.

De plus, Buterin [91] analyse et compare plusieurs méthodes permettant de réaliser des transferts de valeur à travers deux Blockchains différentes. Cette analyse est plus technique que [90] et se focalise davantage sur les principes de fonctionnement de ces méthodes. Plus récemment, Siris et al. [92] comparent également les méthodes permettant d'effectuer des transferts de valeurs ou d'information entre plusieurs Blockchains. Par rapport aux travaux précédents, ils étudient davantage les implémentations pratiques existantes de ces différentes méthodes. Nous allons décrire le fonctionnement et les propriétés de plusieurs de ces méthodes par la suite.

Hashlocking

La méthode de *Hashlocking* est un moyen très naturel d'échanger de la valeur à travers deux chaînes supportant des contrats intelligents simples. L'objectif de ce protocole est de bloquer les fonds de deux utilisateurs dans des contrats qui déverrouilleront les fonds de façon atomique : soit les deux contrats sont déverrouillés, soit les deux contrats restent bloqués.

La FIGURE 2.2 décrit le fonctionnement de la solution de Hashlocking. Les étapes sont les suivantes :

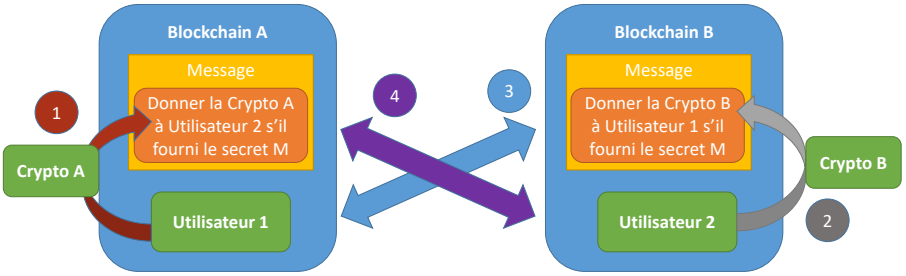

FIG. 2.2 : Schéma descriptif du fonctionnement du Hashlocking.

1. L'utilisateur 1 bloque une certaine quantité de cryptomonnaies A au sein d'un contrat de hashlocking. Le contrat nécessite un secret, et génère un hash. L'utilisateur 1 fournit alors ce hash à l'utilisateur 2.
2. L'utilisateur 2 bloque une certaine quantité de cryptomonnaies B dans un autre contrat de hashlocking qui possède le même hash que le premier.
3. L'utilisateur 1 récupère la cryptomonnaie B en déverrouillant le second contrat. Pour cela, il a besoin de révéler le secret qu'il a utilisé lors de la première étape.
4. L'utilisateur 2 utilise à son tour ce secret pour déverrouiller le premier contrat et récupérer la cryptomonnaie A.

Ce procédé simple possède toutefois deux limitations principales, présentées par Qin et Gervais [93]. La première vient du fait que lors du développement de son contrat de hashlocking, l'utilisateur 2 doit faire attention à un élément : il doit pouvoir récupérer sa monnaie si l'utilisateur 1 ne récupère pas la cryptomonnaie B en un temps imparti. En effet, rien n'oblige l'utilisateur 1 à révéler le secret rapidement, si ce n'est que sa cryptomonnaie A est alors inaccessible. Il y a alors un déséquilibre puisque les deux utilisateurs n'ont pas forcément envie d'utiliser leurs cryptomonnaies au même moment. Pour éviter cela, nous avons besoin d'ajouter deux délais principaux : un délai dans lequel l'utilisateur 1 retire ses cryptomonnaies, et un délai dans lequel l'utilisateur 2 retire les siennes.

Une deuxième limitation est que le nombre de cas d'applications de cette technologie est limité aux transferts de cryptomonnaies. Elle ne permet en effet pas de traiter des changements complexes d'états des deux Blockchains, tels que les exécutions de smart contracts.

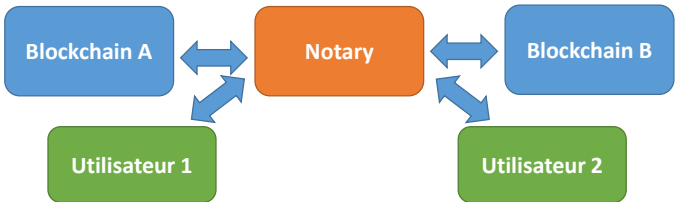

FIG. 2.3 : Schéma descriptif du fonctionnement des Notaries pour le transfert de valeur.

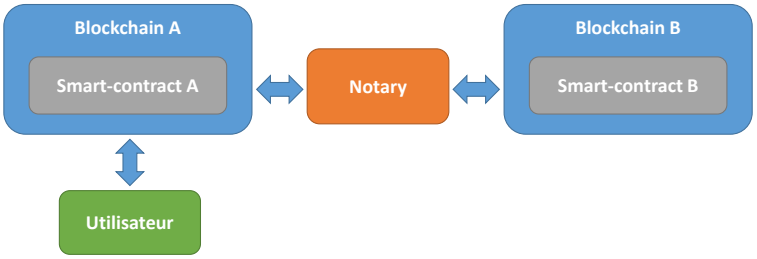

FIG. 2.4 : Schéma descriptif du fonctionnement des Notaries pour l'exécution de smart contracts.

Notaries

Une autre approche simple pour résoudre le problème d'interopérabilité entre Blockchains est d'introduire un intermédiaire de confiance, dont le rôle est de propager l'information d'une chaîne à une autre. C'est le concept de *notaries*, qui est décrit dans la FIGURE 2.3 et la FIGURE 2.4. Nous avons alors l'avantage de pouvoir transmettre des informations arbitraires à travers les chaînes, et pas de simples transferts de valeur.

Cependant, ce système introduit de la centralisation dans des systèmes fondamentalement décentralisés, ce qui limite les cas d'applications, nécessite de faire confiance en des tiers, et créé des *single point of failure* (points individuels de défaillance). Il est possible d'introduire un système de réputation afin de limiter les risques de *notaries* malveillants, mais cela complexifie la méthode.

Bridges et relais

Le principe des *bridges* et relais est similaire à celui des *notaries*, mais sans introduire de tiers de confiance. Cela est accompli en s'assurant que les clients des deux Blockchains que l'on souhaite rendre interopérables puissent lire les données des blocs de l'autre Blockchain. Un exemple de travaux liés aux relais est ETH Relay [94], qui permet de rendre interopérables les Blockchains compatibles avec Ethereum de façon décentralisée.

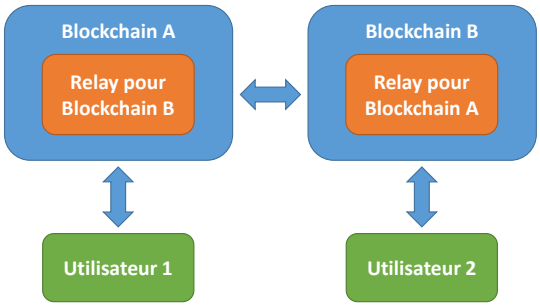

FIG. 2.5 : Schéma descriptif du fonctionnement des Bridges et Relays pour le transfert de valeur.

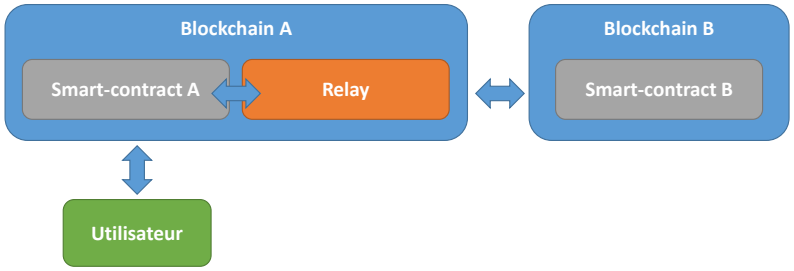

FIG. 2.6 : Schéma descriptif du fonctionnement des Bridges et Relays pour l'exécution de smart contracts.

Cela peut par exemple être réalisé grâce aux concepts de client léger [95]. Un client léger est un client pour une Blockchain qui ne va pas récupérer l'intégralité des données de la Blockchain. Par exemple, les données décrivant chaque transaction ne sont pas stockées ou même téléchargées : seuls les entêtes des blocs sont gardés en mémoire. Ensuite, lorsque l'utilisateur souhaite effectuer une transaction, les données nécessaires à la vérification de la validité des informations liées à cette transaction sont demandées à d'autres clients. Il n'y a alors pas besoin de tiers de confiances puisque les entêtes suffisent à valider l'intégrité des données demandées. Certaines recherches [96] montrent cependant d'importantes limitations sur la sécurité des clients légers sous certaines hypothèses.

Ainsi, les bridges et relais intègrent un client classique pour une certaine Blockchain, et un client léger pour une autre. Il est alors possible de vérifier les informations de cette dernière chaîne et répercuter ces informations sur la chaîne principale. Ce principe est montré en FIGURE 2.5 et en FIGURE 2.6.

Interopérabilité sémantique

Les précédentes sous-sections ont décrit différentes méthodes permettant de faire communiquer des Blockchains différentes. Un des aspects de l'interopérabilité qui peine le plus à évoluer est l'interopérabilité sémantique des Blockchains. En effet, de nombreuses Blockchains ne semblent pas compatibles au niveau fondamental. Un exemple est la différence entre les Blockchains ayant la caractéristique de finalité probabiliste ou certaine. Comme mentionné dans la sous-section 1.2.1, la PoW est un algorithme de consensus qui est intrinsèquement probabiliste. Même lorsque le réseau n'est pas attaqué par des acteurs malicieux, il est possible que deux mineurs créent un bloc valide simultanément. Il est ainsi possible qu'un bloc considéré comme valide soit supprimé ou modifié. C'est ce qu'on appelle une finalité probabiliste.

D'autres algorithmes de consensus ont une finalité certaine, c'est-à-dire que lorsqu'un bloc est considéré comme valide par un nœud du réseau, il ne peut plus être modifié ou supprimé de la Blockchain.

Ces deux types de Blockchains sont alors incompatibles. En effet, si nous souhaitons faire passer des informations d'une Blockchain ayant une finalité probabiliste à une Blockchain ayant une finalité certaine, il est possible de transmettre des informations et de les valider alors qu'elles seront invalidées ultérieurement.

Il existe cependant des moyens probabilistes de lier une chaîne sans finalité avec une chaîne avec finalité, comme ceux présentés dans les publications [48] et [97]. D'un point de vue sémantique, ces recherches permettent ainsi de traduire automatiquement des concepts incompatibles sous hypothèse.

Par ailleurs, certaines terminologies relatives aux contrats intelligents, tels que les graphes de flot de contrôle ou les états, sont formalisées par Chatterjee et al. [98]. Ce type de travaux peut être utile pour interconnecter deux Blockchains supportant les contrats intelligents, mais également pour connecter une Blockchain avec d'autres systèmes interagissant avec des contrats intelligents sur cette Blockchain.

Enfin, d'autres travaux, notamment *BLockchain ONtology with Dynamic Extensibility* (BLONDiE) [76] et *Ethereum Ontology* (EthOn) [77], visent à élaborer des ontologies sur l'écosystème Blockchain. L'écosystème Blockchain est composé par l'ensemble des projets de développement de Blockchain. Cependant, ces projets sont incomplets afin de réaliser des implémentations cohérentes et non triviales de Blockchains sémantiquement compatibles.

La FIGURE 2.7 présente notre proposition d'une ontologie Blockchain simple. Différents concepts, mais également leurs relations, sont définis grâce à OWL. Ici, nous avons modélisé une petite partie des concepts et relations liées à la Blockchain Ethereum. Cette ontologie modélise les éléments suivants :

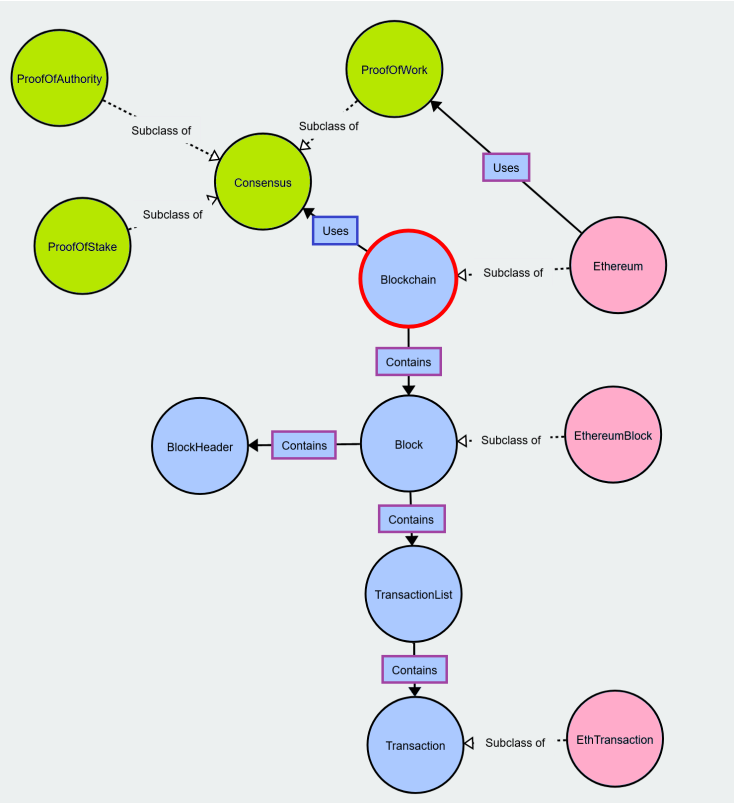

FIG. 2.7 : Exemple d'une ontologie.

- Une Blockchain contient des Blocs,
- Un Bloc contient un entête de bloc et une liste de transactions,
- Une liste de transaction contient des transactions,
- Une Blockchain utilise un algorithme de consensus,
- Enfin, Ethereum est une Blockchain, qui utilise pour algorithme de consensus la PoW.

Sandra [78] est un moteur ontologique créé par la société EverdreamSoft. L'entreprise utilise ce moteur au sein de la librairie Crystal Spark Cannon [79], intégrant des concepts Blockchain tels que les adresses ou les assets collectionnables d'un jeu, les *Non-Fongible Token* (NFT). Les NFT sont des représentations sur la Blockchain d'assets du jeu. Les joueurs peuvent les acheter, vendre ou se les échanger librement. La possession d'un NFT est alors répercutée au sein du jeu.

La principale différence entre Sandra et les deux ontologies précédentes est son utilisation au sein de projets industriels et au sein de la société Everdream-Soft développant l'outil. Ainsi, il est possible d'effectuer simplement des requêtes telles que la récupération de l'ensemble des assets associés à une adresse donnée. L'outil déduit alors du format de l'adresse la Blockchain sur laquelle effectuer la requête et fournit en sortie l'ensemble des éléments trouvés.

2.2.2 Interopérabilité au sein d'une même Blockchain

Pour de nombreuses applications nécessitant d'intégrer la technologie Blockchain, il n'y a pas besoin de concevoir tout un nouveau protocole. Grâce aux contrats intelligents, il est en effet facile d'utiliser une Blockchain déjà existante pour son projet afin de tirer parti des propriétés intrinsèques à la Blockchain, mais également l'écosystème complet de la Blockchain choisie.

Pour ce cas d'interopérabilité, nous allons nous focaliser sur la Blockchain sur laquelle se base le plus grand nombre de projets, Ethereum. Cette Blockchain visant à supporter des contrats intelligents génériques, des applications dans de nombreux domaines l'utilisent, tels que la finance [99] ou les jeux vidéo [7].

Afin que l'ensemble des projets de l'écosystème Blockchain puissent communiquer correctement, il y a besoin d'importants efforts de standardisation. À titre d'exemple, afin d'éviter la volatilité des cryptomonnaies classiques, un projet peut choisir d'intégrer au sein de son application le Dai [43], une monnaie stable par rapport au dollar. Pour cela, le projet à besoin de pouvoir lire et interagir avec le contrat du token Dai.

La standardisation se fait en grande partie par la création d'*Ethereum Improvement Proposal* (EIP) et l'adoption d'*Ethereum Request for Comment* (ERC). Ces processus ont trait au protocole en lui même [100], aux structures de données utilisées au sein de l'écosystème [101], ainsi qu'aux tokens (comme le token ERC-20 [102]) et contrats intelligents. Ainsi, le Dai [43] suit le standard de token ERC-20, c'est-à-dire qu'il suit une interface qui correspond aux fonctions nécessaires à l'utilisation d'une cryptomonnaie. Le Dai peut ainsi être transféré entre utilisateurs et il est possible de demander au contrat le solde en Dai d'un utilisateur. Il est alors facile de l'intégrer dans de nombreux projets de la Blockchain Ethereum. Un exemple d'intégration possible est la gestion du paiement au sein d'une application avec le token Dai.

Groupes de standardisation et consortiums

Cependant, le développement de standards ne provient pas uniquement de l'industrie Blockchain. En effet, des groupes tels que l'*Institute of Electrical and Electronics Engineers* (IEEE) et le NIST, qui développent des standards de nombreuses industries, contribuent également à l'écosystème Blockchain.

Par exemple, l'IEEE Blockchain Initiative [103] et l'IEEE Standards Association [104] proposent des solutions Blockchain, telles qu'un standard de framework pour l'utilisation de la Blockchain au sein de l'Internet des objets [105]. En outre, le NIST indique dans [106] que les objectifs actuels et futurs liés à la standardisation des Blockchains concernent : la terminologie et la taxonomie, l'étude des différents cas d'utilisation de la technologie, ainsi que l'interopérabilité.

D'autres organismes se sont créés spécialement dans le but de créer des standards relatifs à la Blockchain. C'est le cas de l'*Enterprise Ethereum Alliance* (EEA) [107], qui a permis la création d'une architecture standardisée pour des clients Ethereum destinés aux entreprises et Hyperledger, soutenu par la fondation Linux. Comme mentionné 1.2.1, Hyperledger est un incubateur de projets poussé par la fondation Linux. Leurs projets visent à démocratiser l'utilisation de la technologie Blockchain au sein des entreprises. En outre, ils proposent des projets de développement de Blockchains, de librairies, et d'outils pour faciliter le développement d'applications Blockchain au sein des entreprises. Ce sont ainsi les principaux acteurs en termes de développement de Blockchain privées, et se focalisent sur la compatibilité avec les Blockchains publiques existantes, comme Ethereum. C'est pourquoi Hyperledger Fabric, ainsi que Hyperledger Sawtooth et Hyperledger Burrow sont compatibles avec l'*Ethereum Virtual Machine* (EVM) [108], facilitant l'interopérabilité avec Ethereum au niveau des données.

2.2.3 Interopérabilité entre les Blockchains et les autres systèmes

Le dernier niveau d'interopérabilité que nous allons traiter correspond à celle entre les Blockchains et les autres systèmes. En effet, une application décentralisée a par exemple besoin d'une interface utilisateur, qui doit être adaptée au cas d'usage considéré. Il est alors nécessaire de bien comprendre comment l'intégration d'une Blockchain au sein d'une application impacte l'interface utilisateur conçue. Nous étudions ce cas d'interopérabilité en profondeur au sein de la section 2.3, mais nous verrons que l'interopérabilité actuelle de ce niveau d'interopérabilité des systèmes Blockchains reste limitée.

2.2.4 Synthèse de l'Interopérabilité des Systèmes Blockchain

Le domaine de la Blockchain subit depuis sa création de nombreuses avancées. Plus récemment, de nombreux centres de recherches académiques et universités souhaitent formaliser les travaux de recherche liés à la technologie, en particulier sur les aspects d'interopérabilité. L'interopérabilité des systèmes Blockchains se réalise sur trois niveaux. Tout d'abord, entre les Blockchains, des solutions permettent de transférer de la valeur entre différentes Blockchains. Il s'agit du *hashlocking*, des *notaries*, et des *bridges* et relais. De plus, au sein d'une même Blockchain, des travaux de standardisation tels que les ERC, ou les

travaux de groupes au sein de l'IEEE permettent une meilleure interopérabilité des projets étant établis sur une Blockchain commune.

Cependant, il existe encore peu de résultats améliorant l'interopérabilité entre les Blockchains et les composants nécessaires à l'élaboration d'applications entièrement décentralisées. Nous allons décrire dans la prochaine section les limitations des travaux sur l'interopérabilité des DApps.

2.3 Interopérabilité dans le cadre des applications Blockchain décentralisées

L'objectif de cette section est de comparer les différentes solutions d'interopérabilité des DApps. Pour cela, nous commençons par définir les différents éléments à étudier. Nous considérons que les DApps peuvent être étudiées selon quatre aspects : les applications, les données, les processus, et les services.

Du point de vue des DApps en tant qu'applications, l'interopérabilité consiste en la capacité à rendre interopérables deux DApps différentes. Nous ne traiterons pas de ce type d'interopérabilité, pour deux raisons. D'une part, une partie est déjà résolue au sein de la section précédente, puisque l'on cherche en général à faire communiquer deux DApps utilisant la même Blockchain. D'autre part, l'interopérabilité entre deux DApps se réalise également en faisant directement communiquer les services qui composent ces DApps.

Nous appelons données d'une DApp toute donnée qui la compose. Par exemple, il s'agit du code des smart contracts de la DApp, des transactions Blockchains qui assurent le fonctionnement de cette application, ou encore des données utilisateur stockées en dehors de la Blockchain.

Les processus d'une DApp sont l'ensemble des actions et flux d'information à travers les composants de celle-ci. C'est ce que nous modélisons grâce à BPMN comme mentionné dans la section 2.1.1. Nous ne traitons donc pas des processus dans cette section.

Les services d'une DApp sont tous les composants qui permettent de la faire fonctionner. Il peut s'agir d'une Blockchain, d'un service de stockage distribué, d'un service de scalabilité du nombre de transactions que l'on peut effectuer sur la DApp, etc. Nous détaillons au sein de la section 2.4 l'ensemble des services Blockchain utilisables par une DApp.

Ainsi, l'étude de l'interopérabilité dans le cadre des DApps doit considérer l'ensemble de ces points.

2.3.1 Les données et protocoles de communication entre systèmes

L'interopérabilité des données des DApps se réalise entre les services qui composent une DApp.

L'interopérabilité technique liée aux données au sein des DApps est relativement complète dans l'industrie. En effet, l'ensemble des services qui peuvent être utilisés au sein d'une application intégrant la technologie Blockchain propose déjà des API. Par exemple, le portefeuille Ethereum Metamask [109], la solution de scalabilité Matic [110], ou la solution de stockage distribué de données *InterPlanetary File System* (IPFS) [111] proposent des API ou des SDK pour accéder facilement à leurs données et fonctionnalités depuis un autre service.

Cependant, le développement de standards de communication entre ces systèmes permet d'améliorer l'interopérabilité syntaxique liée aux données des DApps. Les différents composants d'une application Blockchain décentralisée doivent communiquer entre eux. Différents projets adoptent des méthodes diverses pour faciliter les transferts d'information entre systèmes.

Tout d'abord Matrix [112] est un protocole définissant différentes API pour aboutir à des communications décentralisées. Ce protocole est utilisé dans diverses applications, et pas seulement des applications Blockchains. D'autres protocoles, comme Devp2p Wire Protocol [113], sont plus spécifiques à la Blockchain.

Enfin, de nombreux chercheurs évitent le problème de transfert de données entre une Blockchain et d'autres systèmes en utilisant sur la Blockchain des références, des hashs, vers leurs données stockées sur d'autres systèmes. Les avantages de cette solution sont d'être simple à mettre en place, de ne pas demander beaucoup de ressources de calculs ou de bande passante, et d'assurer l'intégrité des données.

Cependant, il n'existe pas de manière standardisée permettant un système unique de références. Cela est dommage, puisqu'un système unique de référence permettrait de réaliser simplement des tâches telles que la recherche de ressources par leur hash sur tous les systèmes implémentant ce standard commun.

2.3.2 Étude des DApps par les services

Nous commençons cette section par motiver la représentation des DApps sous forme de services. Pour cela, nous présentons des architectures pour DApps qui se basent sur les services. Ensuite, nous étudions l'intérêt des ontologies pour traiter de l'interopérabilité sémantique des DApps en modélisant les relations entre les différents services d'une DApp.

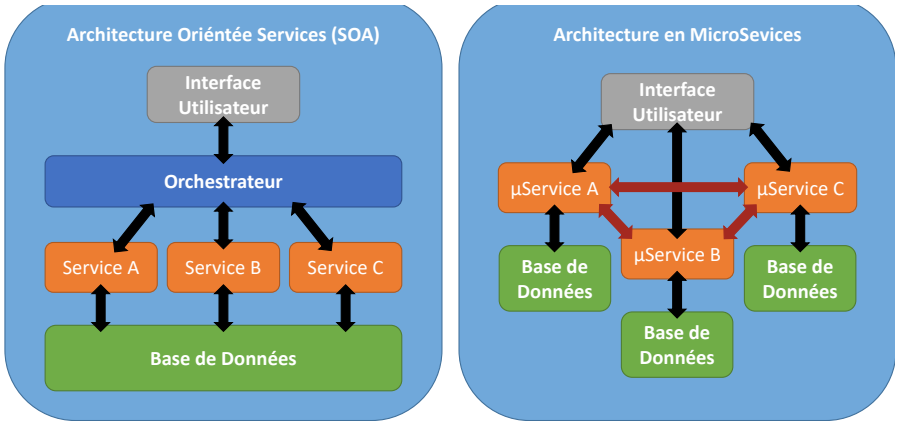

FIG. 2.8 : Les différences entre les SOA et les microservices, inspiré de `https://www.tiempodev.com/blog/microservices-vs-soa`

Concepts liés aux architectures dans l'informatique orientée services

Une approche possible pour concevoir une architecture consiste à composer l'application en services qui exécutent chacun une fonctionnalité de l'application. Nous pouvons différencier deux approches principales dans la conception actuelle des architectures : les architectures orientées services, ou *Service-Oriented Architecture* (SOA) et les architectures microservices.

Selon [114], les microservices sont plus adaptés aux flux de travail décentralisés. En effet, ils soulignent la différence entre l'orchestration de services et la chorégraphie de services. L'orchestration de services se produit lorsqu'un élément central synchronise les différents services d'une application. Au contraire, la chorégraphie de services repose sur le fait qu'un service peut se synchroniser avec les services avec lesquels il interagit directement. Par conséquent, une architecture pour les DApps peut essentiellement s'appuyer sur une approche basée sur les microservices.

La FIGURE 2.8 présente les différences entre ces deux architectures.

Architecture des chaînes de blocs et des technologies de grands livres distribués

Les bases de données traditionnelles ont une gouvernance centralisée et soumise à autorisation. Un utilisateur se voit attribuer un rôle et un niveau d'accès, par exemple administrateur, accès en écriture, accès en lecture, etc.

Au contraire, les Blockchains, ainsi que d'autres types de DLT, ont souvent une gouvernance décentralisée et sans permissions, ou une gouvernance avec permissions mais distribuée. Dans ce cas, le terme distribué fait référence à

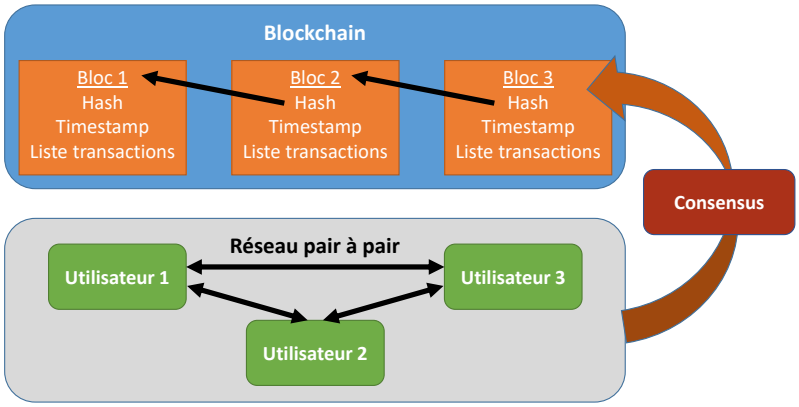

FIG. 2.9 : Exemple d'architecture de DLT.

plusieurs machines effectuant un calcul ensemble, tandis que décentralisé fait référence au contrôle et à la gouvernance du calcul. Les termes sans permission et avec permission font référence aux droits d'accès de chaque machine pour participer au calcul donné.

Cai et al. [115] proposent une architecture générique pour les systèmes Blockchain. Leur architecture est assez simple, avec trois composants : la Blockchain comme structure de données, un réseau pair-à-pair et un modèle de consensus. La FIGURE 2.9 présente leur proposition d'architecture.

Différentes Blockchains et DLT peuvent utiliser diverses méthodes de consensus et avoir différents paramètres de protocole. Cela signifie qu'elles ont chacune des caractéristiques différentes, en termes de fonctionnalités, de mise à l'échelle du nombre de transactions qu'elles traitent, de bande passante des données, de latence, etc. Par souci de simplification, nous choisissons de ne pas prendre en compte ces caractéristiques dans notre méthodologie. Nous choisissons plutôt de concentrer notre travail sur les caractéristiques des autres services, tels que les services de stockage distribué, les interfaces utilisateur ou les services de calcul distribué.

Architecture des Applications Blockchain Décentralisées

L'interopérabilité entre les Blockchains et les autres systèmes est un problème complexe. Le moyen le plus simple de le résoudre est de proposer une architecture générique adaptée aux DApps. En effet, une telle architecture mettra en lumière les besoins d'interfaçage entre les différents composants. Cette approche se rapproche d'une interopérabilité syntaxique entre les sous-systèmes nécessaires.

Les architectures existantes actuellement sont cependant spécifiques à une

application ou un domaine. C'est le cas de ce que propose Hoard dans le jeu vidéo [116], en réalisant des abstractions de concepts comme les moteurs de jeux, ou encore Kim et Jeong [117] pour la gestion de l'authentification des personnes via la Blockchain. Il est également nécessaire d'interagir avec différents moyens de stocker les données de façon décentralisée, comme le montrent Kryukov et Demichev, [118]. Enfin, d'autres travaux montrent le besoin d'inclure des systèmes tiers parfois complexes au sein d'applications Blockchains, tels que les *Trusted Execution Environment* (TEE) [119].

L'un des travaux majeurs pour l'architecture de DApp dans l'industrie a été conçu par IBM : [120]. Ils décrivent une architecture pour les DApps qui convient aux applications d'entreprise. Cependant, elle ne décrit que des systèmes de haut niveau. Ils proposent alors une vue exhaustive des caractéristiques que chaque composant d'une DApp devrait avoir, mais pas la meilleure façon de sélectionner ces caractéristiques.

Nous avions également réalisé une architecture de DApp, en adoptant un modèle par couche. Un schéma représentant graphiquement notre architecture est présenté FIGURE 2.10. Cette proposition d'architecture a fait l'objet d'une publication lors de la conférence intitulée 2019 IEEE International Conference on Blockchain and Cryptocurrency (ICBC2019) [71].

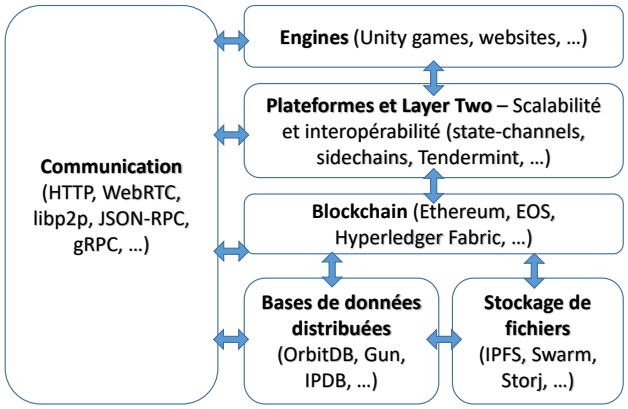

FIG. 2.10 : Exemple d'architecture de DApp, repris de [71]

D'autres travaux sur les architectures de DApp ont une approche plus orientée vers les applications. Par exemple, [121] spécifie une architecture de référence pour les applications *Internet of Things* (IoT) pair-à-pair basées sur la Blockchain. Ils ne ciblent qu'un seul type de cas d'utilisation, mais ils s'attachent à fournir des solutions concrètes aux développeurs d'applications IoT qui souhaitent intégrer la technologie Blockchain. En effet, ils se concentrent sur le

traitement des paiements et de l'identité pour un très grand nombre de dispositifs. Ils ne considèrent pas les applications qui nécessitent de stocker de grandes quantités de données sur la Blockchain, recommandant d'utiliser d'autres canaux de communication tels que UDP ou TCP pour ce faire. En outre, ils recommandent également d'utiliser des Blockchains à autorisation pour les applications IoT en temps réel, car l'utilisation de Blockchains publiques entraîne des problèmes de latence. Ces architectures facilitent la conception et le développement d'une Blockchain. Elles décrivent les services qui devront être mis en œuvre pour une DApp et comment gérer les interactions entre eux.

Cependant, nous constatons certaines limites dans l'utilisation de ces architectures pour concevoir entièrement une DApp. Par exemple, un développeur de DApp pourrait utiliser l'une des architectures décrites précédemment afin de savoir quels services il doit mettre en œuvre pour son application. Cependant, il ne pourra pas en déduire les solutions concrètes à mettre en œuvre.

Ontologies formalisant les Blockchains

Tout d'abord, nous avons vu précédemment que les ontologies sont utiles pour décrire les systèmes Blockchain. En effet, les ontologies visent à définir sémantiquement les différents concepts nécessaires dans un domaine donné. Les systèmes Blockchain impliquent souvent différents domaines qui utilisent des concepts similaires, qui peuvent ne pas avoir de définitions cohérentes. Par conséquent, plusieurs travaux tentent de définir sémantiquement ce qu'est une Blockchain.

Par exemple, BLONDiE [76], FIGURE 2.11, et EthOn [77], FIGURE 2.12, utilisent OWL pour décrire de telles ontologies. Elles sont utiles pour obtenir une compréhension globale de la manière dont les différents concepts de Blockchain tels que les transactions, l'adresse et les signatures sont liés les uns aux autres, ainsi que pour formaliser ces concepts, mais nous n'avons trouvé aucune application utilisant ces ontologies. Cependant, le cadre PHP Sandra [78] permet aux utilisateurs de concevoir facilement leurs ontologies Blockchain, et est utilisé par EverdreamSoft pour interroger les actifs Blockchain.

Grâce à l'utilisation d'ontologies, les différents systèmes qui interagissent avec et dans une DApp ont tous la même définition des concepts et des structures de données utilisés, ce qui améliore l'interopérabilité sémantique d'une DApp.

BLONDiE BLONDiE [76] cherche à formaliser les concepts de base liés aux Blockchains Bitcoin et Ethereum. Cette ontologie présente la Blockchain comme une structure de données uniquement, en détaillant la composition des blocs, et des transactions incluses dans ces blocs.

Un des intérêts de cette ontologie est de formaliser les différences entre ces deux Blockchains. Par exemple, Bitcoin repose sur un modèle de transaction

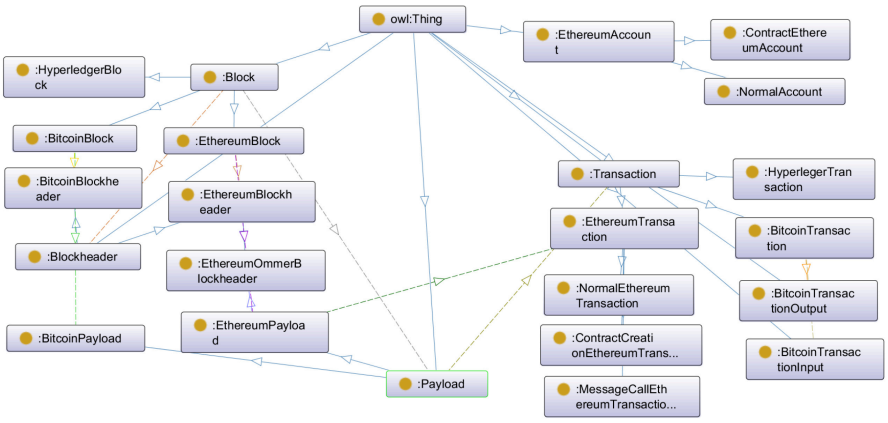

FIG. 2.11 : Graphique représentant l'ontologie Blockchain BLONDiE.

par UTXO, donc BLONDiE définit les concepts de BitcoinTransactionOutput et de BitcoinTransactionInput. Ethereum se basant sur un modèle de compte et solde de compte, ces deux concepts n'ont pas de sens au sein de l'écosystème Ethereum, et ne sont donc pas définis.

En revanche, cette ontologie ne propose aucune formalisation liée au protocole régissant ces deux Blockchains. Il s'agit d'une limitation importante, puisque cela restreint les applications possibles d'une telle ontologie.

EthOn EthOn [77] ne cherche à définir que les concepts liés à la Blockchain Ethereum, mais entre davantage dans les détails du protocole. Par exemple, la relation de minage d'un bloc par un compte Ethereum est définie.

Une des limitations de cette ontologie est se restreindre aux concepts liés à la Blockchain Ethereum elle-même, et ne formalise donc pas l'ensemble de l'écosystème autour de la Blockchain. Par exemple, les différents services Blockchain tels que les solutions de scalabilité de *layer two* ou les portefeuilles de cryptomonnaies ne sont pas modélisés au sein de EthOn.

Concepts relatifs aux services Blockchain Ainsi, les ontologies Blockchain existantes, telles que BLONDiE et EthOn, ne définissent pas les concepts relatifs aux services Blockchain. Pourtant, ces concepts sont fondamentaux pour correctement formaliser une DApp. Nous proposons ainsi au sein du chapitre 4 une ontologie Blockchain centrée sur les DApps et les services Blockchain associés.

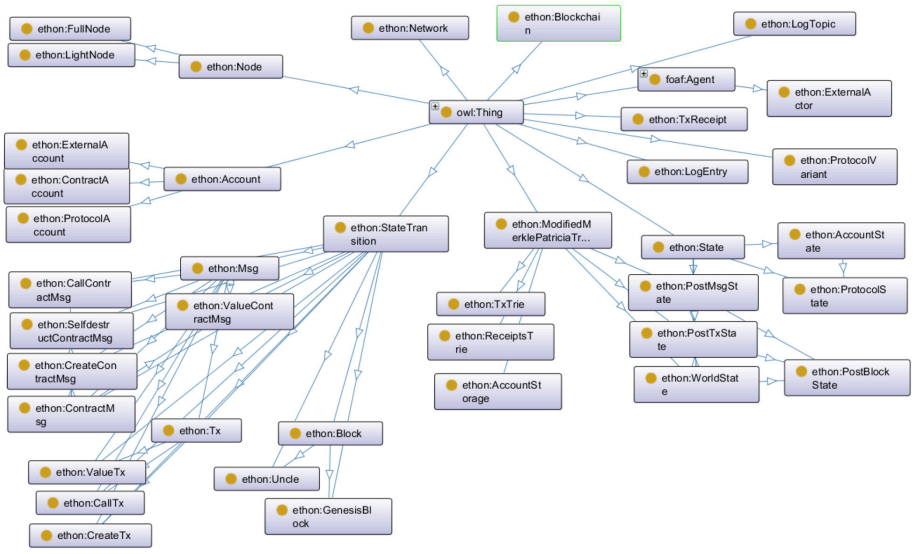

FIG. 2.12 : Graphique représentant l'ontologie Blockchain EthOn.

2.3.3 Synthèse de l'interopérabilité des applications Blockchain décentralisées

Plusieurs travaux industriels et de recherche essayent de résoudre le problème d'interopérabilité au sein des DApps. Cependant, les approches actuelles restent limitées. Par exemple, certains travaux proposent des architectures de DApp, améliorant l'interopérabilité syntaxique de ces systèmes, mais ces architectures sont chacune spécifiques à un cas d'usage particulier, tels que l'IoT ou les applications en entreprise.

De même, certains travaux améliorent l'interopérabilité sémantique au sein des DApps. C'est le cas des ontologies BLONDiE et EthOn. Cependant, ces ontologies ne formalisent pas l'ensemble des concepts nécessaires à la modélisation de DApp et de leurs services, et ont donc des possibilités d'application réduites.

Ainsi, il n'existe pas aujourd'hui d'outil complet aidant à l'interopérabilité des DApps. C'est pourquoi nous proposons au sein du chapitre 3 une méthodologie visant l'amélioration de ce type d'interopérabilité.

2.4 Services Blockchain

Pour nos travaux de recherche, il est important d'avoir un état de l'art des différents services associés à la technologie Blockchain. En effet, cela permettra par la suite de formaliser leurs caractéristiques.

2.4.1 Scalabilité

De nombreuses approches différentes permettent d'augmenter les capacités d'une Blockchain donnée. Ces solutions sont appelées *Layer 2 scalability*, puisque ce sont des sur-couches d'une Blockchain. Ainsi, pour une application fonctionnant sur Ethereum, il est possible d'intégrer des technologies améliorants les caractéristiques de l'application. Nous pouvons distinguer les approches proposées de la façon suivante :

- Les solutions vérifiant la validité des transactions envoyées par le biais de *Validity Proofs*,
- Les solutions demandant des preuves de transactions invalides, appelées *Fraud Proofs*, qui doivent être fournies dans un délai imparti. Ce délai est appelé la période de challenge, comme mentionné par Warren et Bandeali [122].

De plus, nous différencions également :

- Les solutions dont le stockage des données est hors-chaîne,
- Les solutions dont le stockage des données est sur la chaîne.

Finalement, BuildBlockchain Tech et Avihu Levy [123] proposent de classifier les caractéristiques des Layer 2 existantes selon ces deux caractéristiques, comme le montre le tableau 2.1.

Vérification / Données	Validity Proofs	Fraud Proofs
Sur la Blockchain	ZK-rollups	Optimistic rollups
En dehors de la Blockchain	Validium	Plasma

TAB. 2.1 : Table des caractéristiques de différentes solutions de scalabilité de Layer 2, adapté de [123]

State-channels

Les state-channels consistent en l'échange de messages hors-chaîne entre les partis impliqués. En cas de conflit entre ces partis, chacun peut publier sur la Blockchain les messages transmis, et un smart contract vérifiera les informations échangées. Le concept de state-channel est par exemple utilisé par FunFair [124], mais uniquement entre deux participants. Une version schématique de ce concept est présentée FIGURE 2.13.

Des approches plus générales existent, telles que le Lightning Network [125] pour la Blockchain Bitcoin, le Raiden Network pour Ethereum, ainsi que Counterfactual [126]. Cependant, ces approches demandent des recherches supplémentaires. Par exemple, pour le Lightning Network et le Raiden Network, des

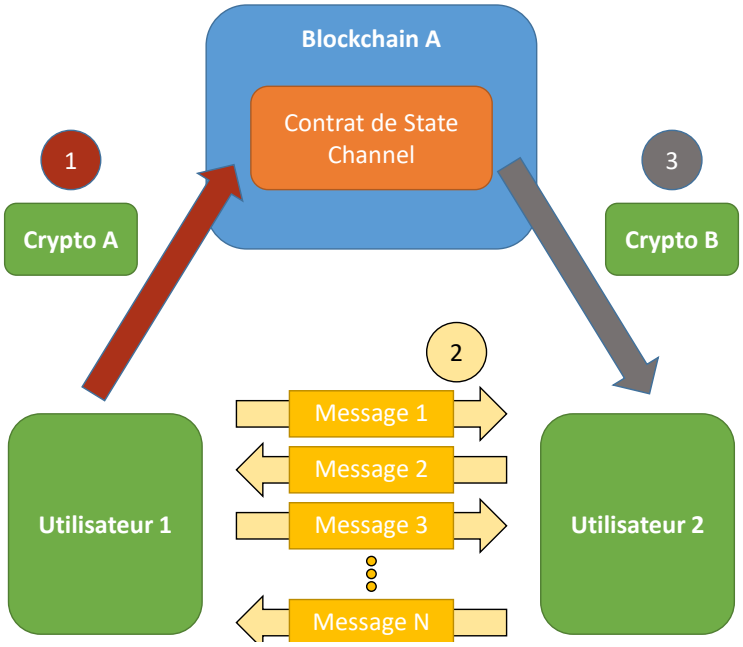

FIG. 2.13 : Schéma explicatif du fonctionnement des State-channels.

recherches sont nécessaires afin de trouver un routing optimal des transactions sur les nœuds qui forment des utilisateurs sur le réseau. La FIGURE 2.14 explique comment ces réseaux peuvent relier deux nœuds par un channel de façon indirecte.

Une des caractéristiques des state-channels est le besoin d'avoir des liquidités pour faire fonctionner le système. Ainsi, il est probable que des *hubs* se créent afin de proposer cette liquidité contre frais.

Rollups

Les *Rollups* [127] sont un type de solution de scalabilité visant à compresser les transactions Blockchains. Pour cela, différentes techniques peuvent être utilisées, comme l'agrégation de transactions en *batchs*. Une autre technique est de stocker une certaine quantité de données de transactions hors-chaîne. Cependant, certaines données de chaque transaction restent sur la Blockchain.

Le concept de Rollups est expliqué FIGURE 2.15. L'agrégation de transactions permet de compresser les données efficacement. Par exemple, une multi-signature, qui est une signature unique valide si toutes les signatures des transactions sous-jacentes sont valides, est moins volumineuse que l'ensemble des signatures individuelles.

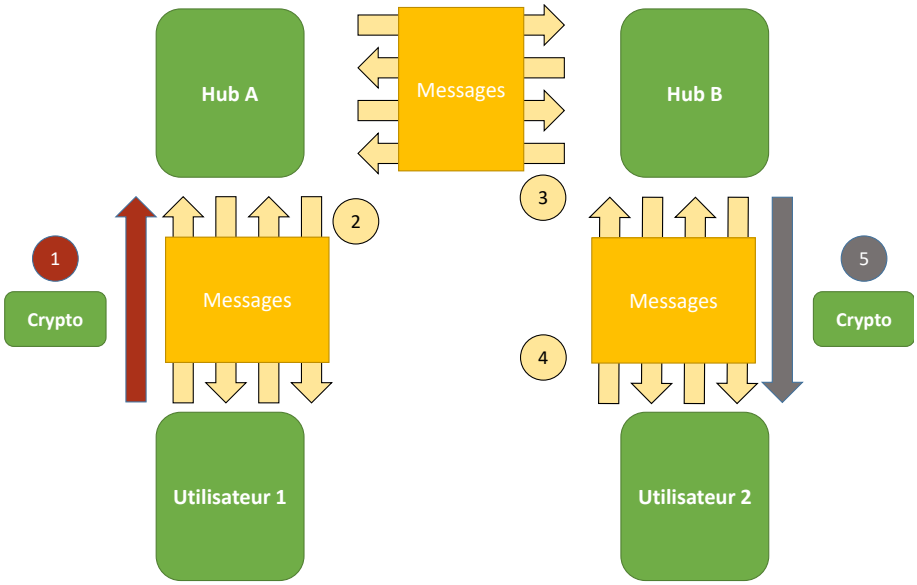

FIG. 2.14 : Schéma explicatif du fonctionnement du Lightning Network et du Raiden Network.

De plus, comme nous allons le voir, deux méthodes principales permettent d'accroître la scalabilité des transactions par rollups : les *Zero-Knowledge Rollups* (ZK-rollups) et les Optimistic rollups.

ZK-rollups Les ZK-rollups, se basent sur l'utilisation de *Zero-Knowledge Proofs* (ZKP). Les ZKP permettent de prouver une information sur des données sans divulguer d'autres informations.

Les ZK-rollups utilisent ce concept afin de compresser les données de transactions, tout en prouvant la validité de la compression. Il s'agit donc de l'utilisation de *Validity Proofs*.

Loopring [128] est un exemple de projet utilisant des ZK-rollups afin d'augmenter le nombre de transactions possibles sur leur place d'échange décentralisée.

Un exemple de projet permettant la scalabilité des DApps est ZKSync [129], une plateforme de Layer Two pour Ethereum qui utilise également les ZK-rollups.

Optimistic rollups Au contraire des ZK-rollups, les Optimistic rollups ne se basent pas sur des preuves formelles de validité des données de transaction compressées. À la place, des *Fraud Proofs* sont utilisées. Cela signifie que si

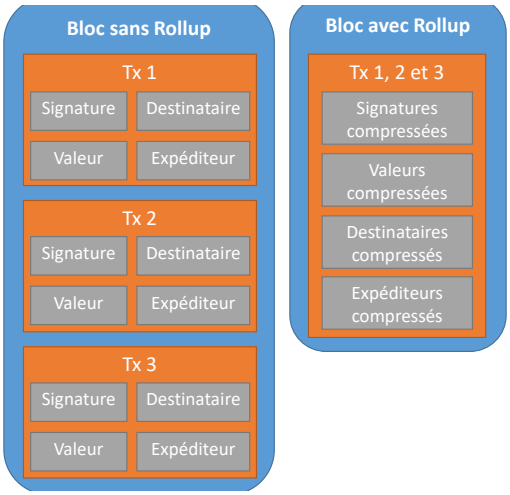

FIG. 2.15 : Schéma explicatif du fonctionnement des Rollups.

aucune preuve de l'invalidité d'une transaction n'est apportée dans un temps imparti, alors cette transaction est déclarée valide.

Un exemple de projet d'optimistic rollups est Arbitrum [130], qui a été lancé le 31/08/2021.

Synthèse sur les rollups Buterin [127] compare quantitativement les caractéristiques de ces deux types de rollups.

En plus des différences entre les deux types de rollups présentés ici, chaque rollup doit choisir certaines caractéristiques qui pourront impacter les propriétés de la rollup. Par exemple, la centralisation, ou non, du processus de soumission de *batchs* de transactions. La centralisation de ce processus permet de le simplifier, mais rend également possible la censure de certaines transactions, puisqu'un seul acteur choisi quelles transactions sont incluses dans le *batch*.

Validium (StarkEx)

StarkEx [131], développé par StarkWare, est une solution de scalabilité basée sur les ZKP. Elle peut être utilisée en tant que ZK-rollups, en gardant les données sur chaîne.

Cependant, StarkEx peut également être utilisé hors-chaîne. Le problème de disponibilité des données est contourné via un *Data Availability Committee*, qui doit assurer la disponibilité de toute donnée incluse dans le Merkle Tree d'un checkpoint.

Nous avions cité Loopring, une place de marché décentralisée utilisant les ZK-rollups. Un de leur concurrent, DeversiFi [132], est basé sur StarkEx, avec l'utilisation de Validium au lieu des ZK-rollups.

Sidechains

Une sidechain est une Blockchain, appelée « Blockchain fille », liée à une autre Blockchain, appelée « Blockchain mère » par un ancrage bidirectionnel (*two-way peg*) [133]. Cette Blockchain est en général utilisée pour une unique application, comme un jeu vidéo. Il est alors possible d'utiliser une méthode de consensus différent de la Blockchain mère, telle que le PoA [31], DPoS [29], ou la PoS [26]. L'avantage d'utiliser une sidechain et non une Blockchain séparée est que si le consensus de la Blockchain fille échoue, les utilisateurs peuvent prouver sur la Blockchain mère ce qu'il s'est passé sur la Blockchain fille. Un design générique de sidechain, appelé Plasma [134], est implémenté dans plusieurs projets. Nous allons nous focaliser sur deux solutions de sidechains existantes, Loom Network [135] et Polygon [136].

Finalement, les sidechains forment une des solutions de scalabilité les plus flexibles pour les développeurs. La FIGURE 2.16 montre le principe de base des sidechains. Ce principe possède quatre étapes principales :

1. Un utilisateur transfère une cryptomonnaie A sur le contrat de dépôt de la sidechain sur la chaîne principale.
2. La sidechain va alors attribuer automatiquement les fonds déposés à l'utilisateur.
3. L'utilisateur peut alors utiliser ces fonds sur la sidechain, à un coût fortement réduit par rapport aux transactions de la Blockchain principale. Régulièrement, un *checkpoint* des derniers blocs de la sidechain est créé et soumis à la Blockchain principale. Ce checkpoint est obtenu en hashant ces blocs, et contribue à la sécurité du système.
4. Finalement, l'utilisateur peut récupérer ses fonds sur la chaîne principale à tout moment. Un certain délai est cependant nécessaire avant que le transfert soit effectif afin de laisser du temps pour les challenges en cas de litiges. En effet, les sidechains fonctionnant par *Fraud Proofs*, si un utilisateur cherche à valider des transactions invalides, un autre utilisateur peut apporter la preuve de la non-validité de ces transactions. Un certain temps doit être accordé pour faire fonctionner ce principe.

Matic Matic [137] est une sidechain liée à Ethereum, fonctionnant par *Fraud Proofs*, et permettant par exemple le *mint* de tokens sur Matic avant de les renvoyer vers Ethereum. Le *mint* est la génération d'un nouvel exemplaire d'un token. Nous détaillons les caractéristiques de Matic au cours de ce mémoire, puisque nous avons étudié en détail son fonctionnement pour valider nos contributions et notre application industrielle.

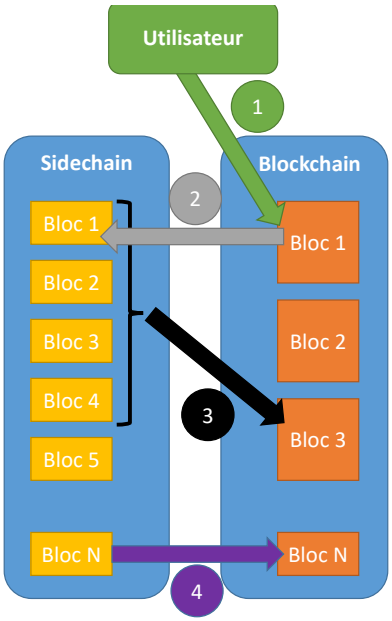

FIG. 2.16 : Schéma explicatif du fonctionnement des Sidechains.

Loom Loom [135] a débuté comme une autre sidechain liée à Ethereum, dédiée principalement au monde du jeu vidéo. Aujourd'hui, ils développent une plateforme multi-chaînes.

Multi-chaînes

Les multi-chaînes sont des solutions de scalabilité utilisant l'interopérabilité entre différentes Blockchains par des Bridges [138]. Comme le montre la FIGURE 2.17, nous pouvons voir en quelque sorte ces solutions comme étant une généralisation des sidechains. Chacune des quatre Blockchains montrées sur ce schéma est en effet liée à une autre, appelée *Hub* ou *Relay Chain* par le même mécanisme d'ancrage bidirectionnel. Les Blockchains « filles » sont appelées des *shards*.

Ainsi, au lieu d'avoir une Blockchain mère liée à une Blockchain fille, un ensemble de Blockchains peuvent être liées ensemble par une couche de validation commune. Cependant, les principes de communication entre les *shards* d'une multi-chaîne varient entre les différentes implémentations de ce principe, que nous allons à présent détailler.

Polygon Polygon [139] est développé par la même équipe que Matic. Ils reprennent et étendent leur solution de sidechain pour créer une multi-chaîne. Ainsi, chaque chaîne faisant partie de Polygon utilise le même système de validation par PoS.

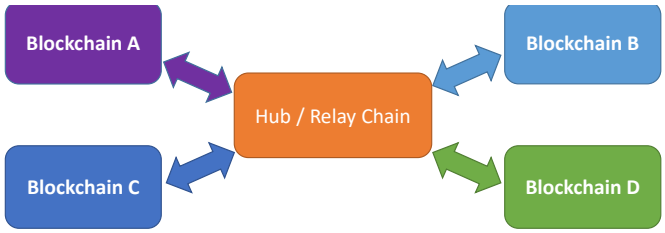

FIG. 2.17 : Représentation schématique du fonctionnement des Multichains.

Polkadot Polkadot [138] propose une multi-chaîne fonctionnant avec leur algorithme de consensus appelé *Nominated Proof of Stake* (NPoS). Cet algorithme est hybride entre la PoS et la DPoS.

Une particularité de Polkadot est que chaque *shard* possède la même sécurité. La sécurité de la multi-chaîne est donc coopérative, ce qui signifie que chaque *shard* possède les mêmes validateurs.

Polkadot propose l'utilisation d'une chaîne principale, appelée Relay Chain, pour faciliter les communications entre les autres chaînes.

Cosmos / Tendermint Cosmos [140] est un projet très similaire à Polkadot, et fonctionne par PoS. La principale différence est sur le modèle de validation des différentes chaînes. Contrairement à Polkadot, Cosmos propose une sécurité compétitive. Ceci signifie que chaque *shard* gère sa propre validation, et doit donc prendre en compte la sécurité des *shards* avec lesquels il communique.

De même que Polkadot propose une *Relay Chain*, Cosmos propose l'utilisation de chaînes particulières appelées *hubs*. Cependant, plusieurs *hubs* différents peuvent être utilisés, et leur utilisation est intrinsèquement nécessaire. En effet, ce sont ces *hubs* qui implémentent les bridges nécessaires entre les différents *shards* de Cosmos.

Harmony Harmony [141] propose une multi-chaîne par sharding, via une validation en PoS. Leur bridge avec Ethereum permet, concrètement, d'utiliser Harmony en tant que multitudes de sidechains liées à Ethereum.

2.4.2 Stockage de données

InterPlanerary File System

InterPlanetary File System (IPFS) [142] est un système de stockage distribué gratuit, fonctionnant en pair-à-pair. Son fonctionnement est simple : tout le monde peut choisir de créer un nœud IPFS sur son système, et y ajouter des fichiers.

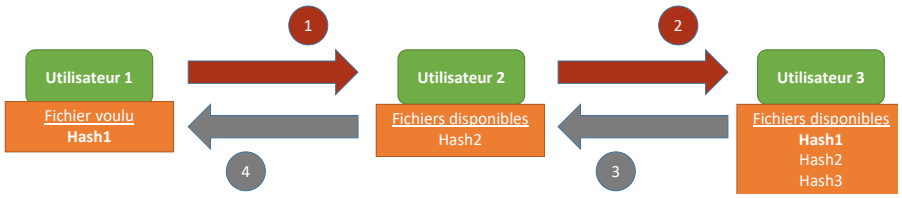

FIG. 2.18 : Schéma explicatif du fonctionnement de IPFS.

Sur IPFS, chaque fichier est représenté par son hash, si bien que n'importe qui peut accéder gratuitement à une ressource stockée sur IPFS par le biais de son hash. Ainsi, pour accéder à une ressource, un nœud IPFS va propager la demande aux nœuds voisins. L'ensemble des nœuds étant un maillage du réseau, après un certain routage, la demande est propagée à un nœud qui possède une ressource possédant le même hash que celle de la demande. Ce nœud transmettra alors la donnée au nœud demandeur.

La FIGURE 2.18 montre comment un utilisateur d'IPFS peut obtenir une ressource qu'il ne possède pas encore, à partir de son simple hash. Dans cet exemple, l'utilisateur 1 recherche un fichier dont le hash est $Hash1$, que seul l'utilisateur 3 possède. Cependant, les utilisateurs 1 et 3 ne sont pas connectés directement, ils doivent communiquer avec l'utilisateur, intermédiaire des requêtes. Les étapes 1 et 2 correspondent à la découverte réseau : les nœuds du réseau propagent les requêtes de demande de fichiers, jusqu'à trouver un utilisateur qui possède le bon fichier. Une fois un client possédant le fichier voulu découvert, les étapes 3 et 4 correspondent à l'envoi du fichier, en parcourant à l'envers le chemin de demande du fichier.

Ce système a des propriétés intéressantes, puisqu'il est gratuit. De surcroît, le risque de perte d'accès à une donnée est mitigé, puisqu'un acteur souhaitant avoir une donnée accessible sur IPFS peut très bien héberger lui-même un nœud IPFS. Cependant, d'autres ressources peuvent devenir inaccessibles, puisqu'aucune contrepartie financière n'est proposée à ceux qui partagent leur capacité de stockage avec le reste du réseau. De plus, la latence de ce système pair-à-pair est assez élevée, 7 secondes en moyenne [143].

Filecoin

Filecoin [144] est une sur-couche à IPFS proposant justement des contreparties financières aux participants. Cette solution peut être vue comme des ventes effectuées sur la Blockchain de capacité de stockage distribué.

Storj

Storj [145] est une autre solution de stockage distribué. Contrairement à IPFS ou Filecoin, il s'apparente davantage à un Cloud décentralisé, puisque les

données y sont répliquées, clusterisées, et chiffrées.

2.4.3 Calcul distribué

Smart contracts

Les smart contracts sont la solution de base au calcul distribué. Ainsi, de nombreuses DApp n'utilisent pas d'outils de calculs distribués additionnels. Les smart contracts permettent l'exécution décentralisée de fonctions déterministes sur une Blockchain. Cette contrainte de déterminisme vient du fait que le résultat de la fonction doit être vérifiable par l'ensemble des nœuds du réseau. En effet, chaque nœud qui reçoit un nouveau bloc de données doit vérifier la validité de chaque transaction de ce bloc, ce qui inclut l'exécution des transactions de smart contract. Elle doit aussi être vérifiable n'importe quand après son exécution. Cependant, des travaux tels que proposés par Chatterjee et al. [146] permettent d'exécuter certaines fonctions normalement impossibles dans un cadre déterministe, comme la génération de nombres pseudo-aléatoires.

Concrètement, les Blockchains supportant les smart contracts, dont la plus importante est Ethereum, proposent des langages de programmation pour écrire ces smart contracts. Leur exécution se fait alors dans une machine virtuelle, telle que l'EVM. [108] Lorsqu'un mineur inclus dans un bloc une transaction représentant un appel de smart contract, il exécute le code dans sa machine virtuelle, puis change l'état de la Blockchain en fonction du résultat. Ce résultat est alors propagé au reste du réseau avec les autres changements d'état qu'amène le bloc créé.

Finalement, en plus de la contrainte de déterminisme, les smart contracts proposent en général des langages limités par la Blockchain. Ainsi, le coût d'exécution est élevé pour des calculs pas nécessairement complexes. Le stockage de données sur la Blockchain étant très cher, cela limite également les possibilités d'entrées ou de sorties de données lourdes.

iExec

iExec [147] est une entreprise visant à développer un marché décentralisé d'achats et vente de puissances de calculs.

La validation du travail effectué par les vendeurs de puissance de calcul est réalisée par redondance. Cela signifie que la tâche est réalisée par différentes personnes indépendantes, qui doivent s'accorder sur le résultat pour le valider.

TrueBit

TrueBit [148] est une plateforme avec un objectif similaire à iExec. La principale différence entre les deux projets est le système de validation du travail effectué.

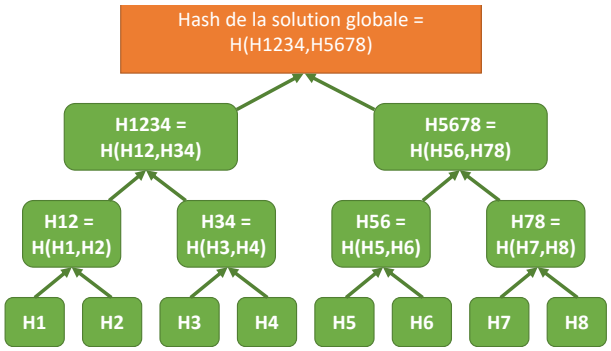

FIG. 2.19 : Schéma explicatif du fonctionnement de Truebit.

Contrairement à iExec, TrueBit fonctionne en effet par *Fraud Proofs* : un travail est découpé par dichotomie en tâches de plus en plus simples, jusqu'à obtenir des tâches élémentaires que l'on sait facilement vérifier directement sur la Blockchain. Ainsi, en cas de conflit sur le travail réalisé, le vendeur peut prouver chaque sous-tâche réalisée, et la personne ayant initié le conflit doit montrer quelle tâche élémentaire pose problème.

Ce principe de fonctionnement du système de validation de TrueBit est détaillé en FIGURE 2.19. Pour chaque opération élémentaire, on calcule localement le hash du changement d'état de la Blockchain. On construit alors un arbre binaire, appelé *Merkle Tree*, dont chaque nœud est obtenu en hashant ses fils. Grâce à ce système, un conflit peut alors être résumé en une opération. Un smart contract se charge alors de calculer le changement d'état de cette opération uniquement, et résout le conflit.

2.4.4 Synthèse des services Blockchain

Finalement, de nombreux services, de différents types, proposent des solutions permettant d'améliorer l'écosystème des DApps. Certains de ces services peuvent changer les caractéristiques de coûts et latence d'une Blockchain. C'est le cas des *sidechains*, des *state-channels*, des *rollups* ou encore des *multichain*. Ces différentes solutions techniques ont leurs avantages et inconvénients, synthétisés au sein du tableau 2.2. De plus, les différentes instances de ces services effectuent des choix d'implémentation différents. C'est par exemple le cas de Loopring et d'Arbitrum. Il s'agit de deux instances de rollups, mais Loopring se base sur les ZKP afin de valider les changements d'état de la Blockchain, tandis qu'Arbitrum se base sur les Fraud Proofs, ce qui signifie qu'en cas de conflits un des acteurs doit proposer une preuve formelle de l'erreur.

D'autres services proposent également de nouvelles fonctionnalités qui ne pourraient pas être possibles via une simple Blockchain, comme le calcul distribué ou le stockage de données distribué.

Méthode	**Avantages**	**Inconvénients**
State-channels	Facilité d'implémentation Coûts négligeables	Nécessite un collatéral Difficultés pour $n \geq 3$ partis
ZK-rollups	Sécurité identique à la Blockchain	Difficulté d'écriture des ZKP
Optimistic Rollups	Facilité d'implémentation	Délais de challenge
Sidechains	Facilité d'implémentation Projets existants aboutis	Délais de challenge
Multichains	Permet l'interopérabilité entre de nombreuses Blockchains	Complexité du système

TAB. 2.2 : Tableau synthétisant quelques avantages et inconvénients des principales méthodes de scalabilité des transactions des Blockchains.

Chacune des solutions présentées possède des propriétés et contraintes différentes, en termes de coûts, sécurité, décentralisation, de maturité ou encore des cas d'applications possibles. Cela signifie que choisir quels services utiliser pour construire une DApp est complexe, et nécessite d'étudier l'ensemble des services disponibles. Ainsi, pour bien comprendre ces propriétés et contraintes, il sera nécessaire dans le prochain chapitre de formaliser leur fonctionnement et leurs contraintes d'utilisation.

Chapitre 3

Développement d'une méthodologie de conception d'application Blockchain décentralisée

Ce chapitre présente une méthodologie qui aide à la conception d'application Blockchain décentralisée. Pour cela, nous proposons des outils et analyses qui permettent la formalisation de telles applications.

Dans le précédent chapitre, nous avons vu différentes limitations des travaux actuels visant une amélioration de l'interopérabilité des DApps. Notre recherche vise à répondre à ses limitations, en proposant en outre une méthodologie d'élaboration de telles applications. Cette méthodologie a fait l'objet d'une publication lors de la conférence intitulée 2020 International Conference on High Performance Computing & Simulation (HPCS2020) [69].

3.1 Motivations

Cette méthodologie, détaillée au sein des sous-sections suivantes, permet de simplifier le développement des DApps. Nous assurons également l'interopérabilité :

- entre les composants d'une DApp,
- ainsi qu'entre les différentes DApp créées à partir de notre méthodologie.

Un aspect essentiel pour les DApps est que pour les cas d'utilisation plus avancés, ces applications ne peuvent pas s'appuyer uniquement sur une Blockchain. Ainsi, les Blockchains apportent de multiples contraintes dans la conception des applications.

Tout d'abord, le partage de données par Blockchain présente une latence élevée. Cette contrainte peut être envisagée de deux manières distinctes.

- Si nous avons seulement besoin que les données soient propagées dans le réseau sans l'immutabilité et la sécurité de la Blockchain, la Blockchain n'ajoutera aucune latence en plus des transferts de données distribués. Dans ce cas, la latence est plus élevée que dans un système centralisé, mais elle convient à la plupart des applications et peut être modélisée avec précision.

- Cependant, si nous avons besoin que les données soient immuables, nous devons attendre un certain nombre de confirmations de blocs pour que la probabilité d'un retour en arrière devienne négligeable. Ici, le temps de confirmation dépend de l'algorithme de consensus sélectionné, car, par exemple, la PoW [25] confirme les blocs plus lentement que la PoS [26] ou que la DPoS [29]. Le partage des données est également affecté par l'utilisation du réseau, et d'autres paramètres tels que la taille du bloc, et enfin la sécurité dont on a besoin.

Les Blockchains ont également une faible bande passante pour le partage des données. Cela signifie que les Blockchains ne sont pas adaptées aux cas d'utilisation nécessitant un nombre élevé de transactions, ni aux cas d'utilisation nécessitant le traitement d'une grande quantité de données.

Un cas d'utilisation particulièrement inadapté aux Blockchains devrait donc être les jeux vidéo, car les jeux vidéo multijoueurs ont généralement besoin d'une faible latence pour être en temps réel, d'un taux de communication élevé entre les joueurs et de gros fichiers images à transférer. Ce cas d'utilisation sera détaillé dans le chapitre 6.1, mais il montre la nécessité d'inclure d'autres composants dans la conception des DApps.

Pour résoudre ce problème de partage de données, un développeur de DApp peut choisir de concevoir son application selon des patrons de conception qui peuvent être regroupés en deux catégories principales :

- L'utilisation de systèmes semi-centralisés pour éviter d'utiliser la technologie Blockchain pour des tâches qui ne sont pas adaptées aux Blockchains. Par exemple, la plupart des DApps existantes, telles que CryptoKitties [7], ont leur code *front-end* stocké sur des pages web centralisées. Dans le cas de CryptoKitties, les images des actifs virtuels sont servies par les serveurs de l'entreprise, ce qui entraîne un point de défaillance unique. Bien que la définition des DApps implique qu'ils ne s'appuient que sur des systèmes

décentralisés, cette solution de compromis semble être acceptée par la communauté. Malheureusement, ce point de défaillance unique peut conduire à des attaques telles que le détournement de DNS [149].

- L'utilisation des solutions existantes de mise à l'échelle décentralisée. Par exemple, il est possible de stocker une application à page unique, ou *Single-Page Application* (SPA) sur une solution de stockage distribuée telle que IPFS [150]. Un exemple de DApp qui utilise IPFS pour réaliser un jeu entièrement décentralisé est réalisé par EtherPlay [151].

Ces approches habituelles de la conception des IPFS ont prouvé leur efficacité, mais les architectures qui en résultent sont rarement adaptées à de multiples applications. Il en résulte une faible interopérabilité entre les DApps existantes, ce qui réduit leur utilité.

À notre avis, il y a un fort besoin de développer une méthodologie qui, étant donné un cas d'utilisation spécifique pour une application Blockchain, conduit à une architecture DApp adaptée à cette application. Notre approche utilise des ontologies et BPMN pour rassembler les contraintes de l'application et définir l'ensemble des instances de service nécessaires pour la DApp. À notre connaissance, aucun autre système existant ne propose une méthodologie similaire à la nôtre.

Notre méthodologie vise à formuler un ensemble de recommandations pour faciliter le développement d'une DApp. Par exemple, pour chaque interaction entre les utilisateurs de la DApp, une recommandation est fournie pour savoir s'il est considéré comme préférable d'utiliser uniquement une Blockchain pour mettre en œuvre une certaine fonctionnalité de la DApp ou s'il est nécessaire d'inclure d'autres services. Notre méthodologie commence par une liste d'exigences fonctionnelles [152] pour l'application du client, à partir de laquelle les spécifications sont tirées (FIGURE 3.1).

Ces spécifications sont actuellement écrites en langage naturel. Du point de vue de la sécurité, notre méthodologie doit également garantir que les avantages de la technologie Blockchain en matière de sécurité se transfèrent bien à l'ensemble de l'application. Ceci est réalisé en concentrant nos modèles sur les interactions possibles entre tous les acteurs d'une DApp donnée.

3.2 Vue d'ensemble

Compte tenu d'une DApp cible, la méthodologie se divise en étapes suivantes, présentées dans la FIGURE 3.1 :

1. Formaliser à la fois les concepts liés à l'aspect Blockchain de la DApp et ceux liés au domaine d'application. Nous pouvons réutiliser une ontologie existante ou, si nécessaire, nous pouvons étendre une ontologie

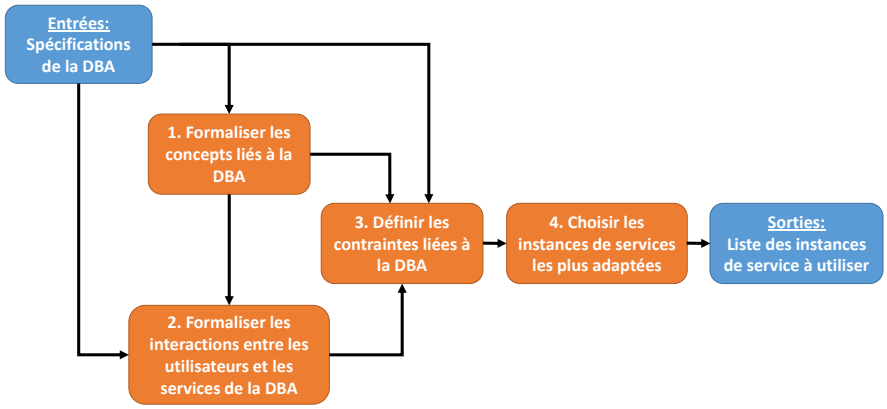

FIG. 3.1 : Vue globale des quatre étapes de notre méthodologie, repris de [69]

existante pour définir de manière exhaustive les concepts de notre DApp. Par exemple, dans notre article [69], nous étendons l'ontologie existante EthOn [77] avec des concepts liés au jeu d'échecs.

2. Formaliser toutes les interactions qui se produisent dans la DApp, entre les services de la DApp ou les utilisateurs. Nous utilisons la modélisation BPMN basée sur les événements pour cette étape : l'événement de départ correspond à une action effectuée par un utilisateur ou un service au sein de la DApp. Il s'agit par exemple du déploiement de la DApp, ou d'une transaction effectuée par un utilisateur. Ensuite, notre modèle montre toutes les communications entre utilisateurs et services qui découlent de cet événement de départ.

3. Définir les contraintes techniques de la DApp. Ici, l'ontologie et les modèles BPMN sont utiles pour obtenir une liste complète de ces contraintes. En effet, l'ontologie peut aider à identifier les contraintes liées à un concept en particulier. Par exemple, si l'ontologie montre que le concept de compte de la Blockchain est critique pour l'application, le modèle de compte de la Blockchain utilisée pourrait conduire à une contrainte technique. Un modèle de sortie de transaction non dépensée, ou *Unspent Transaction Output* (UTXO), qui exige que chaque transaction, à l'exception des transactions de coinbase qui rémunèrent les mineurs, fasse référence à une transaction existante pour être valide, pourrait être préféré à un modèle de compte/bilan plus simple, qui ne garde trace que de l'état associé à chaque compte [153]. De même, pour définir les contraintes liées aux interactions entre les multiples concepts de la DApp, il faut analyser chaque interaction présentée dans les modèles BPMN.

4. Définir les instances de service appropriées pour la DApp. Pour chaque contrainte définie à l'étape précédente, nous choisissons la meilleure ins-

tance de service pour y répondre en utilisant les critères suivants : sécurité de la solution, exigences de décentralisation, nombre de transactions à traiter, volume de données à échanger, ou encore simplicité de mise en œuvre. Le résultat de notre méthodologie est obtenu en faisant correspondre les caractéristiques de chaque interaction dans une DApp avec les fonctionnalités des services supportés par notre méthodologie.

3.3 Détails de la méthodologie

Cette section permet de préciser certains points particuliers des éléments de notre méthodologie présentés précédemment.

3.3.1 Formalisation des concepts de la DApp

La formalisation des concepts de la DApp voulue est nécessaire pour s'assurer que tous ces concepts soient correctement pris en compte par les différents services implémentés. Par exemple, lorsqu'un développeur définit formellement un jeu vidéo Blockchain, il est possible d'en déduire la quantité de données qui doit être stockée par la DApp. Si le plateau de jeu contient des données volumineuses pour calculer l'état actuel d'une partie, alors il faut prendre en compte cette caractéristique dans la conception de la DApp.

Les ontologies, définies avec OWL, permettent de définir formellement ces concepts et leurs relations. La première étape de notre méthodologie consiste donc à la création d'une ontologie détaillant l'ensemble de ces concepts et leurs relations. Cet outil permet aussi de garantir l'interopérabilité sémantique entre les différents services de la DApp.

Afin de faciliter l'application de notre méthodologie, nous proposons dans la suite de nos travaux une ontologie Blockchain. Cette ontologie formalise les concepts généraux liés aux Blockchains et aux DApp, mais propose également les modélisations de certains Blockchain Patterns couramment utilisés par l'industrie Blockchain [154]. Nous étudierons en détail les différents Blockchain Patterns et leur formalisation au sein de la section 4.2.

Ainsi, les chercheurs et développeurs souhaitant appliquer notre méthodologie pour concevoir leurs applications peuvent étendre notre ontologie pour l'adapter à leur cas d'application. Nous détaillons dans les deux chapitres suivants la construction de cette ontologie et son utilisation.

3.3.2 Formalisation des interactions de la DApp

La deuxième étape de la méthodologie vise à formaliser l'ensemble des interactions de la DApp. Cette étape est fondamentale, puisqu'elle permet de s'assurer de la bonne propagation des informations à travers la DApp, et donc

de leur cohérence. Par exemple, lorsqu'un joueur d'un jeu vidéo Blockchain réalise une action, cette action doit être prise en compte en modifiant l'état de la partie, en vérifiant les conditions de victoires, ainsi qu'en étant propagée aux autres joueurs.

L'outil de modélisation BPMN est adapté à ce type de formalisation, puisqu'il permet de décrire l'ensemble des processus qui permettent le fonctionnement de la DApp. Ce formalisme permet par exemple l'utilisation d'événements et s'adapte donc bien aux différentes actions asynchrones possibles au sein d'une DApp. D'autre part, BPMN est un outil générique adapté à l'ensemble des DApps. Son utilisation permet ainsi d'améliorer l'interopérabilité des processus entre ces DApps. Par exemple, plusieurs DApps peuvent réutiliser le même processus de création de comptes afin de faciliter la création de services universels de création de comptes pour les DApps.

La réalisation de modèles de l'application souhaitée par BPMN sert ainsi à s'assurer que toutes les interactions possibles de l'application sont considérées. Une DApp peut avoir besoin de plusieurs modèles BPMN différents afin de formaliser toutes les interactions possibles. Par exemple, pour la modélisation d'un jeu vidéo Blockchain, le processus de création de comptes utilisateur est distinct du processus d'une partie. Il faut donc modéliser ces deux processus par deux BPMN différents.

Nous présentons des exemples de BPMN d'une DApp dans le chapitre 6 détaillant l'application de nos recherches au domaine du jeu vidéo.

3.3.3 Contraintes techniques de la DApp

L'étape de définition des contraintes techniques de la DApp permet de traduire les informations issues des deux étapes précédentes et les spécifications de la DApp en éléments concrets permettant la bonne conception de l'application.

Comme énoncé précédemment, un exemple de déduction de contraintes à partir d'une ontologie est la formalisation de l'état d'une partie. À partir de cette formalisation, le développeur de la DApp peut déduire la contrainte de taille des données à stocker sur la Blockchain ou sur un service de stockage de données.

D'autre part, il est aussi possible de déduire des contraintes depuis les interactions définies au sein des BPMN produits lors de la deuxième étape. Par exemple, pour un BPMN modélisant les conditions de victoires d'un jeu vidéo Blockchain, le développeur peut déduire exactement quelles informations sont nécessaires à leurs évaluations. Une contrainte sur ces informations est donc qu'elles doivent être synchronisées et cohérentes avant l'évaluation des conditions de victoires, ce qui donne lieu à des contraintes de latence pour l'application.

Les contraintes techniques à définir peuvent être de différents types, que nous présentons dans cette section.

Contraintes externes

Les contraintes externes correspondent à un besoin d'un acteur extérieur à l'application. Elles proviennent souvent de contraintes fonctionnelles et sont ainsi plutôt faciles à décrire. Par exemple, dans la grande majorité des cas, un utilisateur de DApp aura besoin d'avoir un compte Blockchain associé à la DApp, ainsi que de la cryptomonnaie pour payer les éventuels frais de transactions.

Contraintes internes

Les contraintes internes correspondent à un détail d'implémentation d'un des composants de la DApp. Un exemple de ce type de contrainte pourrait être la nécessité d'utiliser des smart contracts implémentant un standard particulier, tel que le standard ERC-721.

Contraintes d'intégration

Les contraintes d'intégrations correspondent à une liaison entre deux composants internes à l'application, ou bien entre un composant de l'application et un acteur extérieur. Nous retrouvons ainsi ici les contraintes de coûts et de latence des différentes interactions. Une autre contrainte d'intégration classique est le besoin d'authentifier un utilisateur au sein de l'application. Généralement, un utilisateur est authentifié via son compte Blockchain. Mais ce compte peut être obtenu par la graine du portefeuille, ou par un service tiers tel que Metamask.

Contraintes globales

Les contraintes globales s'appliquent à l'ensemble de la DApp, et impacteront fortement son développement. Par exemple, les coûts de production de la DApp, en prenant en compte le développement et le déploiement des smart contracts, font partie de ce type de contraintes.

3.3.4 Choix des instances de services

La dernière étape de notre méthodologie consiste à choisir, en fonction des contraintes de la DApp, les instances de services à utiliser au sein de l'application. La section 2.4 décrit le fonctionnement de plusieurs services Blockchain qu'un développeur peut choisir d'inclure dans sa DApp.

Ce choix est fait de manière manuelle, en comparant les contraintes de l'application avec les caractéristiques de chaque instance de service disponible.

À titre d'exemple, nous appliquons notre méthodologie à une DApp qui est un registre d'images et de leurs métadonnées. L'objectif de cette DApp est qu'un

utilisateur puisse ajouter des images et consulter l'ensemble des images du registre. Dans ce cas, les coûts impliqués empêchent le stockage des images sur la Blockchain, nous utilisons donc IPFS pour cette tâche. Les autorisations d'accès pour la DApp sont gérées par des contrats intelligents sur la Blockchain Ethereum, car l'application ne nécessite pas d'un environnement d'exécution à haut débit et à faible latence. Il est alors possible de stocker certaines métadonnées, comme le créateur de l'image ou des tags, dans le smart contract. D'autres métadonnées peuvent être plus lourdes, et il faut les stocker sur IPFS. C'est par exemple le cas d'un fichier de licence qui détaille l'ensemble des utilisations possibles de l'image. Enfin, l'interface utilisateur pour le stockage et la récupération des images est une petite SPA, interagissant avec un smart contract Ethereum en utilisant un fournisseur Web3 tel que Metamask.

Une autre possibilité est l'utilisation des travaux de thèse de Ahmed Nacer [155]. Il propose effectivement une méthodologie de sélection de services basée sur une architecture de microservices. Cette méthodologie utilise l'approche de la recommandation multicritères qui consiste à agréger un ensemble de contraintes au sein d'une seule métrique pour classifier les différents choix. En particulier, il utilise un agrégateur appelé *Ordered weighted averaging aggregation operator* afin de classer les instances de services disponibles.

3.4 Synthèse sur notre méthodologie

Nous avons présenté une méthodologie nouvelle de conception de DApp. Cette méthodologie repose sur des outils de formalisation tels que les ontologies et les BPMN afin d'obtenir une liste de services Blockchain adaptés à une application donnée. En effet, une formalisation des concepts liés à la DApp et des interactions au sein de cette DApp permet de s'assurer que l'ensemble des contraintes, fonctionnelles et techniques, de la DApp sont bien prises en compte. Cette formalisation permet par la suite d'étudier plus facilement quels services Blockchains sont les plus adaptés pour l'application cible.

Contrairement aux propositions analogues telles que [82], notre méthodologie se repose sur l'utilisation de BPMN afin de modéliser les processus d'une application. Même si ce type de modèles ne permet pas de représenter en détail les différents composants logiciels d'une application, elle permet de modéliser correctement les interactions au sein de l'application.

Modéliser une DApp via BPMN demande une approche similaire pour l'ensemble des applications. En revanche, concevoir une ontologie formalisant l'ensemble des concepts d'une DApp peut être complexe. C'est pourquoi nous devons proposer une ontologie générique qui permet la formalisation simple de n'importe qu'elle DApp. La construction de cette ontologie est détaillée au sein du chapitre suivant.

Chapitre 4

Développement d'une ontologie Blockchain

Ce chapitre présente les différentes étapes de développement d'une ontologie Blockchain. Cette ontologie permet de modéliser différentes applications Blockchain décentralisées, et ainsi améliorer leur interopérabilité sémantique.

4.1 Élaboration de notre propre ontologie Blockchain

Nous avons décrit dans le chapitre précédent l'intérêt d'avoir une formalisation des différents concepts d'une DApp au sein d'une ontologie. Afin de faciliter la construction d'une ontologie spécifique à une application, nous proposons une ontologie Blockchain générique. Cette ontologie pourra en effet ensuite être étendue lorsque l'on souhaite formaliser une DApp. Dans cette ontologie, nous utilisons la terminologie *Decentralized Blockchain Applications* (DBA) pour formaliser les DApps afin d'éviter les ambiguïtés entre ces applications et les applications décentralisées qui n'intègrent pas la technologie Blockchain.

4.1.1 Vision globale de l'ontologie

Avec l'aide du Plugin OntoGraph pour protégé, un graphique partiel de l'ontologie est présenté FIGURE 4.1. Dans ce graphique, nous pouvons voir un des concepts fondamentaux liés aux Blockchains : les transactions. Une transaction Blockchain est un message signé par un utilisateur, qui sera ajouté à un bloc de la Blockchain. Une transaction peut être *pending*, c'est-à-dire en attente de

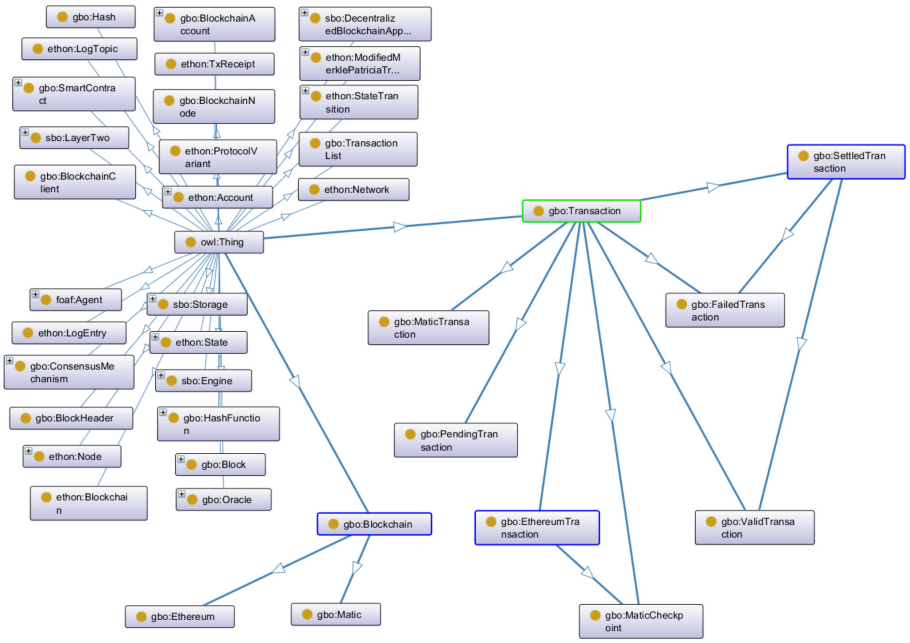

FIG. 4.1 : Visualisation partielle de notre ontologie. Seuls les concepts de Transaction et de Blockchain sont détaillés par souci de lisibilité.

validation, ou *settled*, c'est-à-dire validée ou invalidée. Une fois validée, une transaction ne peut plus être retirée de la Blockchain. Un autre concept montré dans cette figure est la Blockchain, dont Matic et Ethereum héritent. Par souci de lisibilité, nous ne détaillons pas dans cette figure l'ensemble des autres concepts de l'ontologie, tels que les DApps, les solutions de scalabilité des transactions ou encore les méthodes de consensus. Cependant, des figures détaillant des parties restreintes de l'ontologie sont présentées par la suite.

Au sein de la section 2.3.2, nous présentons deux ontologies Blockchain existantes, BLONDiE et EthOn. Afin de construire notre ontologie, nous étudions la possibilité d'étendre ces ontologies existantes.

L'ontologie EthOn formalise de nombreux concepts liés à la Blockchain Ethereum. Par exemple, elle formalise les concepts de transactions Ethereum et les blocs d'Ethereum, mais aussi les appels de contrats et leur création, ainsi que les changements d'état de la Blockchain, et plus généralement les structures de données utilisées par Ethereum et leurs utilisations. Nous avons donc fait le choix d'étendre l'ontologie EthOn, puisqu'elle possède déjà de nombreux concepts importants pour la formalisation des DApps.

BLONDiE est l'autre ontologie Blockchain existante. Elle est intéressante

puisqu'elle formalise différentes Blockchains : Ethereum, Bitcoin, et la Blockchain privée Hyperledger Fabric. Cependant, les concepts présentés ne sont pas assez détaillés. Les transactions de chaque Blockchain sont définies, mais peu d'informations relatives à leur utilisation au sein de la Blockchain sont présentes. Par exemple, les appels ou créations de smart contracts Ethereum sont formalisés comme c'est le cas au sein de EthOn, mais pas les structures de données utilisées pour chaque Blockchain. Ainsi, le concept de *Merkle-Patricia Trie*, qui défini l'arbre stockant l'ensemble des transactions et les changements d'état associés, est présent dans EthOn mais pas dans BLONDiE. Pour cette raison, nous n'étendons pas l'ontologie BLONDiE. En revanche, nous avons tout de même utilisé cette ontologie pour comprendre comment les concepts de différentes Blockchains peuvent coexister au sein d'une même ontologie.

L'ontologie proposée utilise les préfixes suivants :

- ethon, pour l'ontologie EthOn importée.
- gbo, pour formaliser les concepts génériques de la Blockchain. Avec l'aide du Plugin OntoGraph pour protégé, un graphique montrant la partie générique de l'ontologie est présenté FIGURE 4.2.
- sbo, pour formaliser les concepts spécifiques aux DApp. Un graphique montrant la partie spécifique aux DApp de l'ontologie est présenté FIGURE 4.3.
- bpo, pour formaliser différents Blockchain Patterns. La partie dédiée aux Blockchain Patterns de l'ontologie est présentée FIGURE 4.4. Nous listerons par la suite l'ensemble des Blockchain Patterns que nous avons formalisé, ainsi que des exemples de leur application.

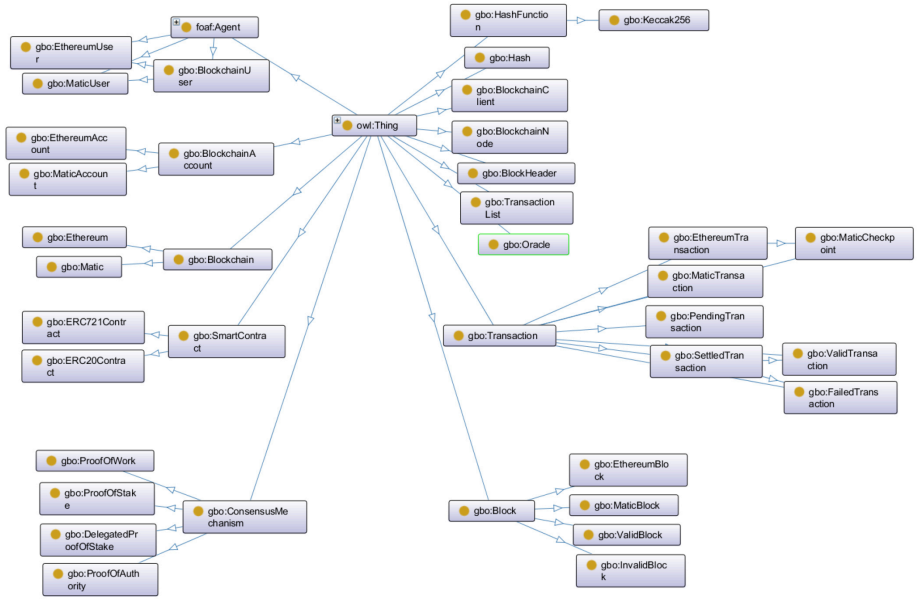

FIG. 4.2 : Visualisation de la partie générique de notre ontologie, comprenant les principaux concepts relatifs aux Blockchains

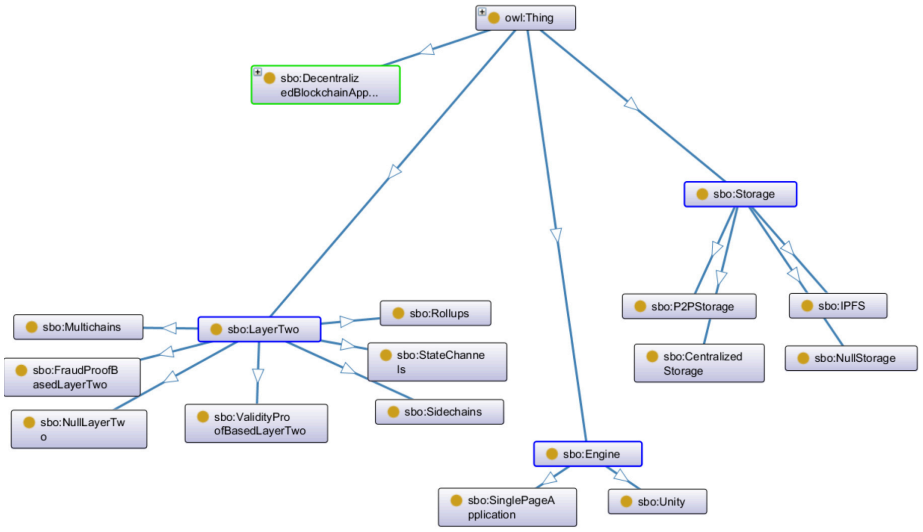

FIG. 4.3 : Visualisation de la partie spécifique aux DApp de notre ontologie.

4.1.2 Classes

Cette sous-section détaille l'ensemble des classes principales de notre ontologie. Chaque classe représente un concept lié à la Blockchain ou aux DApp, et peut être décrite par une annotation, des contraintes, ainsi qu'avec des individus exemples de cette classe. Les contraintes de chaque concept peuvent être liées à un autre concept de l'ontologie, ou non. Dans ce premier cas, l'ontologie représentera la contrainte par le biais de propriétés d'objets ou *ObjectProperties*. Sinon, l'ontologie la représentera par le biais d'une propriété de données ou *DataProperties*.

gbo:Blockchain

Annotation D'après le NIST, une Blockchain est un livre de registre collaboratif et résistante aux tentatives de falsifications, qui maintient des données transactionnelles [14].

Contraintes Les contraintes liées au concept de Blockchain représentent l'ensemble des données et des classes en relation avec ce concept. Concrètement, ces contraintes donneront lieu dans notre ontologie à des DataProperties et des ObjectProperties.

- Utilise une Méthode de consensus, décrite dans la classe gbo:ConsensusMechanism
- Contient des Blocs, décris dans la classe gbo:Block
- Temps moyen de création d'un bloc (secondes), le temps moyen entre chaque bloc
- Taille de bloc (octet), la taille maximale de chaque bloc
- Possibilité de smart contracts (0/1)
- Finalité (secondes, temps de confirmation estimé pour un seuil de certitude fixé)
- Complexité des opérations (si possibilité de smart contracts)

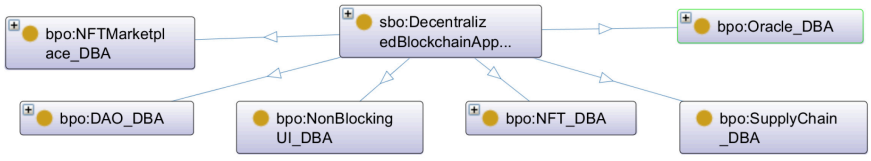

FIG. 4.4 : Visualisation de la partie dédiée aux Blockchain Patterns de notre ontologie.

- Coût d'une opération (€/op)

Les contraintes listées ici correspondent soit à la Blockchain en tant que structure de données (par exemple la taille des blocs), soit aux caractéristiques du protocole (par exemple la finalité).

Exemples d'individus possibles

- Bitcoin : PoW, 360s, 4Mo, 0, 3600s, N/A, 0.10€ - 10€
- Ethereum : PoW, 14s, 1Mo, 1, 300s, N/A, 0.05€ - 5€
- EOS : DPoS, 360s, 4Mo, 0, 3600s, N/A, 0.01€
- HyperLedger Fabric : PoA, 360s, 4Mo, 0, 3600s, N/A, 0€

gbo:Block

Annotation Un bloc est l'unité atomique des changements d'état au sein de la Blockchain. Il est lié cryptographiquement au bloc suivant par son hash.

Contraintes

- Contient une liste de transactions, dont le concept est formalisé dans la classe gbo:Transaction
- Taille d'un bloc (octet)

Exemples d'individus possibles

- Le premier bloc d'une Blockchain est spécial puisqu'il ne peut pas référencer de blocs précédents. Il s'appelle le *Genesis block*.

gbo:Transaction

Annotation Les transactions validées sont présentes dans un bloc. Avant d'être incluse dans un bloc, une transaction est dite « en attente de validation », et appartient à une *mempool*.

Exemples d'individus possibles

- Pour chaque bloc, une transaction particulière est appelée la *coinbase transaction*. Il s'agit d'une transaction qui rémunère le mineur de ce bloc. Ce type de transaction est le seul qui permet une création monétaire.
- La sidechain Matic a besoin de réaliser de *checkpoints* réguliers sur Ethereum. Ces checkpoints sont des transactions sur Ethereum, et permettent aux transactions Matic incluses dans le checkpoint d'obtenir la même sécurité que si elles étaient des transactions classiques sur Ethereum. Ethereum utilisant un algorithme de consensus plus sécurisé que Matic, ces checkpoints sont donc fondamentaux au bon fonctionnement de la sidechain.

gbo:BlockchainUser

Annotation Un utilisateur d'une Blockchain est un humain qui possède et gère un compte sur cette Blockchain. L'accès à ce compte peut être réalisé via un service tiers tel que Metamask, ou bien directement par l'utilisateur s'il possède un nœud de la Blockchain. Un utilisateur peut être un possesseur de cryptomonnaies, un utilisateur de DApp, ou encore un mineur. Un mineur, pour une Blockchain fonctionnant par PoW, participe à la création de nouveaux blocs, sécurise la Blockchain, et est rémunéré pour cette tâche.

gbo:ConsensusMechanism

Annotation Un mécanisme de consensus est un algorithme permettant de résoudre le Problème des Généraux Byzantins. En particulier, il est utilisé pour définir quels blocs et quelles transactions seront considérés comme valides par le réseau.

Exemples d'individus possibles

- Proof of Work
- Proof of Stake
- Delegated Proof of Stake
- Proof of Authority

gbo:SmartContract

Annotation La section 1.2.1 décrit l'objectif des contrats intelligents sur une Blockchain. Dans notre ontologie, un contrat intelligent est utilisé par une DApp et est inclus dans une Blockchain.

sbo:DecentralizedBlockchainApplication

Annotation Comme précisé au sein de la section 1.2.1, une DApp est une application qui utilise la technologie Blockchain. Cependant, la plupart des DApps existantes ont besoin d'autres services associés pour fonctionner, comme un *engine* qui pourra agir comme un *front-end* pour l'application, des solutions de stockage, ou encore des solutions de scalabilité de la Blockchain utilisée.

Dans le chapitre suivant, nous définirons de nombreuses règles SWRL qui permettront de déduire les propriétés d'une DApp à partir de ses caractéristiques et des technologies qu'elle utilise.

Contraintes

- Utilise une gbo:Blockchain
- Utilise au moins un gbo:SmartContract
- Utilise un moteur tel que défini par la suite via la classe sbo:Engine
- Utilise un sbo:LayerTwo tel que défini dans la suite
- Utilise un sbo:Storage tel que défini dans la suite
- Nombre de transactions à réaliser pour le fonctionnement
- Coûts de fonctionnement
- Latences de fonctionnement
- Quantité de stockage de données nécessaire au fonctionnement

Nous détaillerons l'ensemble des DataProperties relatives à cette classe au sein de la section 4.1.5.

sbo:Engine

Annotation Les moteurs, ou *engines*, sont un concept générique lié aux DApp. Selon la DApp considérée, le moteur associé peut être une page web, un moteur de jeu, un framework de gestion d'application, etc. Nous définissons les sbo:Engine comme étant des outils permettant de développer des interfaces pour une DApp.

sbo:LayerTwo

Annotation Nous avons présenté en détail le fonctionnement de nombreuses solutions de scalabilité de niveau deux au sein de la section 2.4.1.

Exemples d'individus possibles

- Matic
- *Zero-Knowledge Rollups* (ZK-rollups)
- Validium

sbo:Storage

Annotation De même, nous avons présenté le fonctionnement de solutions de stockages pour DApp au sein de la section 2.4.2.

Exemples d'individus possibles

- Stockage centralisé
- IPFS
- Filecoin

gbo:MaticCheckpoint

Annotation Un checkpoint Matic est une transaction Ethereum qui permet de valider des transactions Matic. Un nouveau checkpoint est soumis toutes les 30 minutes afin de synchroniser Matic avec Ethereum.

Contraintes

- Contient une liste de transactions Matic validées par ce checkpoint, référencés par leur hash.

4.1.3 Propriétés d'objets

Les propriétés d'objets, ou ObjectProperties sont des relations entre deux des concepts définis au sein de l'ontologie. Ici, nous définissons plusieurs propriétés, relatives à la Blockchain en temps que structure de données (par exemple gbo:Contains ou gbo:IncludedIn), ou bien aux usages fonctionnels d'acteur ou services (par exemple gbo:Uses).

gbo:Contains

La relation gbo:Contains met en relation deux objets lorsque l'un contient l'autre, du point de vue des structures de données. Ainsi, une Blockchain contient des blocs, et les blocs contiennent des transactions.

gbo:IncludedIn

La relation gbo:IncludedIn est l'inverse de la relation gbo:Contains. Ainsi, un bloc valide est inclus dans une Blockchain.

gbo:Uses

La relation gbo:Uses est plutôt générique. Elle peut représenter un acteur qui utilise un système. Par exemple, un utilisateur Blockchain va utiliser une Blockchain.

Cette relation convient également pour indiquer qu'une technologie à besoin d'une autre pour fonctionner. Par exemple, une Blockchain utilise une méthode de consensus, et une DApp utilise une Blockchain.

gbo:**NextBlock**

La relation gbo:NextBlock est utilisée pour référencer le prochain bloc qui est miné par une Blockchain à un certain temps.

gbo:**LastCheckpoint**

La relation gbo:LastCheckpoint permet à une Blockchain qui utilise un système de checkpoints sur une autre de référencer le dernier checkpoint créé. Par exemple, la sidechain d'Ethereum Matic publie régulièrement des checkpoints sur la Blockchain Ethereum.

gbo:**MinesFor**

La relation gbo:MinesFor relie un utilisateur Blockchain à une Blockchain où il est mineur. Cette relation ne peut qu'impliquer des Blockchains possédant des mineurs, donc utilisant la PoW comme méthode de consensus.

gbo:**HasBeneficiary**

La relation gbo:HasBeneficiary relie un bloc valide d'une Blockchain au mineur qui l'a miné.

4.1.4 Individus par classe

Dans cette sous-section, nous définissons les individus dont nous aurons besoin pour formaliser correctement une DApp complète. En particulier, ces individus seront nécessaires pour appliquer les règles SWRL que nous définirons en section 5.1.

gbo:**Blockchain**

sbo:**_MaticMainnet** Cet individu représente la Blockchain Matic principale.

sbo:**_EthereumMainnet** Cet individu représente la Blockchain Ethereum principale, dans sa version actuelle (ETH 1.0).

sbo:**_EthereumRopstenTestnet** Cet individu représente une version de test de la Blockchain Ethereum. Cette version, appelée Ropsten, fonctionne par PoW, comme la chaîne principale, et reproduit ainsi plutôt bien le comportement de la chaîne principale. Comme pour les autres testnets, la principale différence avec le mainnet Ethereum est que la cryptomonnaie de Ropsten (les Ropsten ETH) n'a pas de valeur financière. Ainsi, il n'est pas possible de tester ou simuler les caractéristiques économiques d'une DApp avec un testnet. En effet, la cryptomonnaie n'ayant pas de valeur financière, un acteur de la DApp qui aurait un intérêt à agir d'une certaine façon sur le mainnet risque de ne pas se comporter de la même manière sur le testnet.

gbo:**Engine**

sbo:**_Unity** Cet individu représente un front-end créé par le moteur de jeu Unity. Il est par exemple utile pour le jeu LTR de B2Expand, puisque ce jeu doit appeler des contrats intelligents sur Ethereum depuis un exécutable créé avec Unity.

sbo:**_SPA** Cet individu représente un front-end créé avec des technologies du web telles que React ou Angular. Ce sont des SPA. Cet exemple nous est utile pour modéliser notre DApp de gestion de *Non-Fongible Token* (NFT) développée pour le jeu LTR, présenté au sein de la section 6.2.3. Comme expliqué précédemment, les NFT sont des représentations sur la Blockchain d'assets du jeu. Les joueurs peuvent les acheter, vendre ou se les échanger librement. La possession d'un NFT est alors répercutée au sein du jeu.

gbo:**LayerTwo**

sbo:**_MaticLayerTwo** Cet individu est très similaire à sbo:_MaticMainnet. La différence est qu'il est ici considéré comme une solution de scalabilité pour Ethereum, et non plus une simple Blockchain. Les propriétés associées seront donc différentes.

sbo:**_NullLayerTwo** Cet individu permet de considérer une DApp qui ne possède pas de Layer Two. Il sera utile lors de l'écriture de règles SWRL, en section 5.1.

sbo:**_Multichain** Cet individu représente une solution de scalabilité en multichain.

sbo:**_Rollup** Cet individu représente une solution de scalabilité via rollups.

gbo:**Storage**

sbo:**_CentralizedStorage** Cet individu représente un stockage centralisé, tel qu'un site internet classique.

sbo:**_IPFS** Cet individu représente le stockage distribué IPFS.

sbo:**_NullStorage** Cet individu permet de considérer une DApp qui ne possède pas de stockage dédié. Il sera utile lors de l'écriture de règles SWRL, en section 5.1.

sbo:**_P2PStorage** Cet individu représente le stockage distribué de type pair-à-pair. Cela signifie que les utilisateurs de la DApp se partagent directement les fichiers de la DApp entre eux.

gbo:**DecentralizedBlockchainApplications**

Nous avons détaillé trois cas différents de DApp utiles pour B2Expand. Les trois individus suivants représentent des DApps ayant le même objectif, celui de la gestion des assets de LTR. Cependant, elles utilisent des technologies différentes. L'objectif sera alors de comparer les caractéristiques de ces individus en fonction des technologies utilisées.

Dans les trois cas, le stockage est centralisé, et le moteur est Unity.

sbo:**_DBA1** Cet individu représente une DApp utilisant uniquement Ethereum, sans solution de scalabilité en Layer Two.

sbo:**_DBA2** Cet individu représente une DApp utilisant Ethereum, ainsi que sa sidechain Matic.

sbo:**_DBA3** Cet individu représente une DApp utilisant uniquement Matic. Ici, Matic est donc utilisé en tant que Blockchain et non en tant que sidechain d'Ethereum.

gbo:**MaticTransaction**

Nous avons défini trois individus qui représentent des transactions Matic soumises à des temps différents.

sbo:**_MaticTransaction1** Cette transaction est soumis à la Blockchain Matic le 18/10/2021 à 18h20.

sbo:**_MaticTransaction2** Cette transaction est soumis à la Blockchain Matic le 18/10/2021 à 18h45.

sbo:**_MaticTransaction3** Cette transaction est soumis à la Blockchain Matic le 18/10/2021 à 18h50.

gbo:**MaticCheckpoint**

Nous avons également défini deux individus qui représentent deux checkpoints Matic successifs.

sbo:**_MaticCheckpoint1** Cette transaction est soumise à la Blockchain Ethereum le 18/10/2021 à 18h30.

sbo:**_MaticCheckpoint2** Cette transaction est soumise à la Blockchain Ethereum le 18/10/2021 à 19h00.

4.1.5 Propriétés de données

Les propriétés de données, ou DataProperties, sont des propriétés qui lient une classe avec des données de base. Ces données peuvent être des entiers, des chaînes de caractère, des dates, etc.

Nous avons choisi de formaliser les propriétés de données principales pour quelques concepts, ainsi que celles dont nous avons besoin pour appliquer les règles SWRL dans la section 5.1.

Les DataProperties spécifiques à une DApp sont divisées en quatre catégories : les coûts, les nombres de transactions, les quantités de données, et les latences.

Dans un premier temps, les données *Cost* permettent de formaliser les différents coûts d'une application ou des services qui la composent. Nous avons choisi de différencier les coûts payés par l'utilisateur des coûts payés par le développeur de la DApp. En effet, certains développeurs peuvent préférer réaliser certaines transactions à la place des utilisateurs de la DApp. Cela améliore l'expérience utilisateur, puisque les utilisateurs n'ont pas à payer les frais de transactions pour certaines fonctionnalités de l'application. Par exemple, de nombreux échanges de cryptomonnaie payent les frais de transaction de retrait d'une cryptomonnaie sur un portefeuille de l'utilisateur. Les coûts de stockage de données et de transactions sur une Blockchain sont également considérés.

- Cost : Les propriétés de coûts
    - CostDevPerUser : Coût développeur par utilisateur
    - CostUser : Coût pour chaque utilisateur
    - CostPerMb : Coût d'un mégaoctet de stockage
    - CostTx : Coût d'une transaction
        - CostBasicTx : Coût d'une transaction de transfert de cryptomonnaie
        - CostTxTokens : Coût d'une transaction de transfert de tokens ERC-20, les appels de smart contracts étant plus coûteux que le transfert de cryptomonnaie
        - CostTxDeploy : Coût d'une transaction de déploiement d'un smart contrat

Les données $NbTx$ formalisent, pour chaque DApp, une estimation du nombre de transactions Blockchain à réaliser pour un fonctionnement normal de l'application. Il s'agit d'une estimation puisqu'il est difficile de prévoir précisément l'utilisation que les différents utilisateurs auront de l'application. De même que pour la différenciation des coûts développeur et utilisateurs, nous différencions les transactions effectuées par les utilisateurs de celles effectuées par les développeurs de l'application. Comme énoncé précédemment, dans certains cas,

une fonctionnalité d'une DApp sera uniquement appelée par les développeurs. D'autres fonctionnalités, comme la création et l'achat d'assets ERC-721 sur Ethereum, demanderont plusieurs transactions, réalisées par le développeur et les utilisateurs.

Un des objectifs de cette formalisation étant de formaliser les coûts d'une application, nous différencions également les transactions qui doivent être réalisées sur une Blockchain, de celles qui peuvent être réalisées sur un service de scalabilité de type Layer Two. En effet, les coûts des transactions associées à ces services peuvent être très différents. Il est donc important de connaître les contraintes sur les transactions pour modéliser au mieux l'ensemble des coûts.

- NbTx : Les propriétés relatives au nombre de transactions nécessaires au fonctionnement de l'application
    - NbTxDevPerUser : Nombre de transactions que le développeur doit réaliser pour chaque utilisateur
        - NbTxDevPerUserOnMainchain : Nombre de transactions sur la Blockchain principale que le développeur doit réaliser pour chaque utilisateur
        - NbTxDevPerUserOnLayerTwo : Nombre de transactions sur la plateforme de Layer two que le développeur doit réaliser pour chaque utilisateur
    - NbTxPerUser : Nombre de transactions pour chaque utilisateur
        - NbTxPerUserOnMainchain : Nombre de transactions sur la Blockchain principale pour chaque utilisateur
        - NbTxPerUserOnLayerTwo : Nombre de transactions sur la plateforme de Layer two pour chaque utilisateur

De façon analogue aux nombres de transactions, les propriétés *NbMb* ont pour but de modéliser les quantités de données nécessaires au fonctionnement de l'application. De même que précédemment, nous différencions les données stockées par le développeur et par les utilisateurs. En effet, dans la majorité des applications, certaines données, telles que les comptes utilisateur, sont stockées par les développeurs, tandis que d'autres données, comme un exécutable de l'application elle-même, sont stockées en local sur les machines des utilisateurs. Nous différencions également les données sur la Blockchain de celles pouvant être stockées en dehors de cette Blockchain.

- NbMb : Les propriétés de données relatives aux quantités de données à stockées pour un bon fonctionnement de l'application
    - NbMbDevPerUser : Quantité de données à stocker par le développeur pour chaque utilisateur
        - NbMbDevPerUserOnMainchain : Quantité de données à stocker par le développeur sur la Blockchain principale pour chaque utilisateur

        - NbMbDevPerUserOnLayerTwo : Quantité de données à stocker par le développeur sur la plateforme de Layer two pour chaque utilisateur
        - NbMbDevPerUserOnStorage : Quantité de données à stocker par le développeur sur la solution de stockage choisie pour chaque utilisateur
    - NbMbPerUser : Quantité de données à stocker pour chaque utilisateur
        - NbMbPerUserOnMainchain : Quantité de données à stocker sur la Blockchain principale pour chaque utilisateur
        - NbMbPerUserOnLayerTwo : Quantité de données à stocker sur la plateforme de Layer two pour chaque utilisateur
        - NbMbPerUserOnStorage : Quantité de données à stocker sur la solution de stockage choisie pour chaque utilisateur

Enfin, le dernier type de DataProperties correspond aux notions de latence d'une application. Les lectures sur une Blockchain ne nécessitant pas l'application de l'algorithme de consensus, il est donc important de différencier les latences en lecture ou en écriture de la DApp et des services qui la composent.

- Latency : Les propriétés relatives aux latences
    - LatencyReads : Latence en lecture
        - LatencyReadsStorage : Latence en lecture de la solution de stockage utilisée
        - LatencyReadsTx : Latence en lecture d'une transaction
    - LatencyWrites : Latence en écriture
        - LatencyWritesStorage : Latence en écriture de la solution de stockage utilisée
        - LatencyWritesTx : Latence en écriture d'une transaction

4.2 Extension de l'ontologie avec les Blockchain Patterns

Afin de montrer la possibilité de modéliser, grâce à notre ontologie, des cas d'usages variés, nous avons décidé de modéliser différents design patterns existants dans l'industrie Blockchain. Ces patterns correspondent soit à des types de cas d'application de la Blockchain, soit à des méthodes de conception d'un élément d'une DApp. Le terme Blockchain Pattern est repris par Xu et al. [154] pour étudier les *design patterns* utilisés dans l'industrie Blockchain.

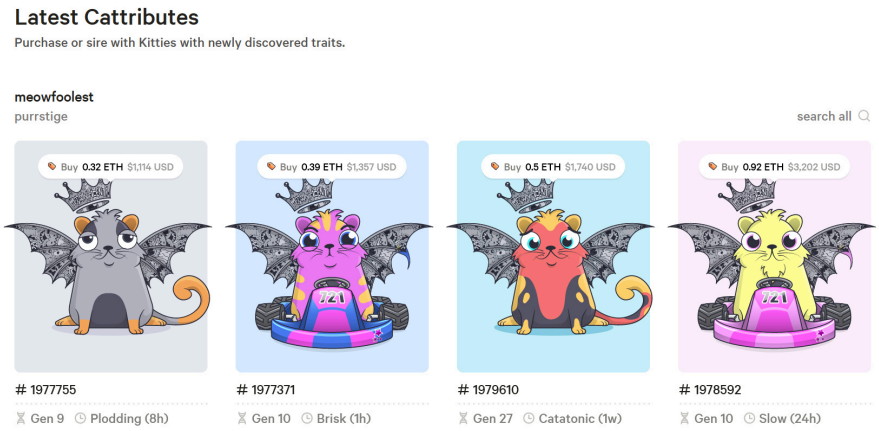

FIG. 4.5 : Exemple d'une DApp gestionnaire de NFT, repris de `https://www.cryptokitties.co`

4.2.1 Application de gestion de NFT

Un grand nombre des DApps existantes visent à gérer des NFT, symbolisés en règle générale par des tokens ERC-721. La plupart des DApps de gestion de NFT, telle que celle développée par B2Expand, créent un nouvel NFT en modifiant l'état d'un contrat existant.

Cependant, certains projets, comme Mintable [156], proposent aux créateurs de NFT deux autres alternatives. Tout d'abord, ils ont déployé un smart contract permettant la création gratuite de NFT, appelée *Gasless minting*. Le NFT créé se retrouve bien sur la Blockchain, mais le coût de la modification de l'état du contrat est payé par l'acheteur d'un NFT, uniquement lors de son achat. Par ce système, un vendeur n'a pas besoin de payer pour la création d'assets non vendus.

De plus, ils proposent un autre contrat permettant de déployer facilement un contrat de vente, et ce même pour un unique NFT. Cela permet, en outre, une plus grande personnalisation de chaque NFT, qui augmente cependant considérablement les coûts.

Un exemple de DApp qui suit ce Blockchain Pattern est CryptoKitties [7], présenté FIGURE 4.5. Cette DApp donne la possibilité aux joueurs de collectionner des chats virtuels. Ces chats sont des NFT, et les joueurs peuvent les acheter, vendre, ou encore les faire se reproduire. Chaque CryptoKitty possède un ensemble d'attributs, lui conférant une valeur financière plus ou moins élevée.

Les DataProperties spécifiques aux DApp de gestion de NFT sont les suivantes :

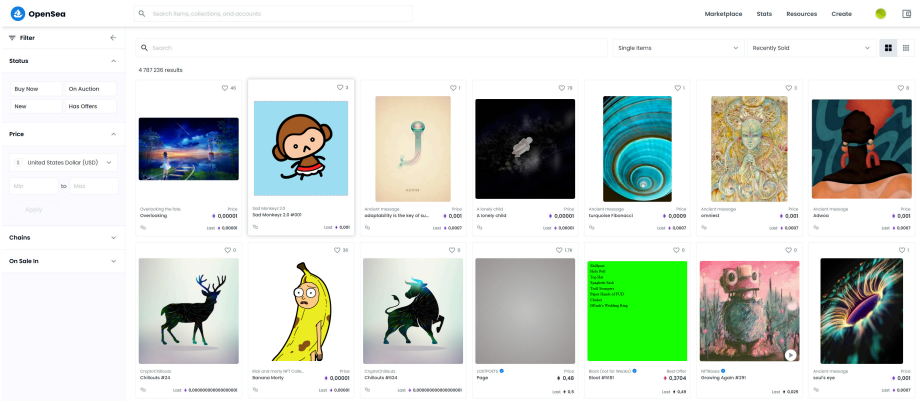

FIG. 4.6 : Exemple d'une DApp Marketplace de NFT, `https://opensea.io`

- DeployContractForEachNFT : Défini s'il faut déployer un smart contract pour chaque NFT ou non
- GaslessMinting : Défini si la création des NFT est gratuite ou non

Ces deux propriétés sont suffisantes pour décrire les caractéristiques fondamentales d'une telle application. Elles permettent en particulier de formaliser les coûts de création d'assets. Une autre propriété importante pour une telle DApp est la solution de stockage utilisée pour les images des assets. Cryptokitties utilise un stockage centralisé des métadonnées des NFT, comme la majorité des applications de gestion de NFT. Cependant, cette propriété n'est pas exclue à de telles applications, et notre ontologie propose la formalisation de la solution de stockage utilisée pour toutes les DApps par le biais de la classe sbo:Storage.

4.2.2 NFT Marketplace

Les NFT Marketplace, tels que OpenSea [157], proposent à leurs utilisateurs l'achat et vente de NFT. La FIGURE 4.6 montre l'interface utilisateur d'OpenSea.

Sur cette application, les vendeurs sont majoritairement des entreprises qui vendent leurs NFT, ou bien des utilisateurs du marché secondaire. Ils vont en moyenne générer un grand nombre de transactions de création et vente de NFT sur la plateforme.

À l'opposé, les acheteurs effectuent un nombre de transactions bas : peu d'entre eux achètent des centaines de NFT différents. Il peut alors être intéressant de différencier dans la modélisation de la DApp les deux types d'utilisateurs.

Les DataProperties spécifiques aux NFT Marketplace sont les suivantes :

- UserBuyer : Propriétés relatives à un acheteur de NFT
    - NbTxPerUserBuyer : Nombre de transactions pour chaque utilisateur acheteur
        - NbTxPerUserBuyerOnMainchain : Nombre de transactions sur la Blockchain principale pour chaque utilisateur acheteur
        - NbTxPerUserBuyerOnLayerTwo : Nombre de transactions sur la plateforme de Layer two pour chaque utilisateur acheteur
    - NbMbPerUserBuyer : Quantité de données à stocker par le développeur pour chaque utilisateur acheteur
        - NbMbPerUserBuyerOnMainchain : Quantité de données à stocker par le développeur sur la Blockchain principale pour chaque utilisateur acheteur
        - NbMbPerUserBuyerOnLayerTwo : Quantité de données à stocker par le développeur sur la plateforme de Layer two pour chaque utilisateur acheteur
        - NbMbPerUserBuyerOnStorage : Quantité de données à stocker par le développeur sur la solution de stockage choisie pour chaque utilisateur acheteur
- UserSeller : Propriétés relatives à un vendeur de NFT
    - NbTxPerUserSeller : Nombre de transactions pour chaque utilisateur vendeur
        - NbTxPerUserSellerOnMainchain : Nombre de transactions sur la Blockchain principale pour chaque utilisateur vendeur
        - NbTxPerUserSellerOnLayerTwo : Nombre de transactions sur la plateforme de Layer two pour chaque utilisateur vendeur
    - NbMbPerUserSeller : Quantité de données à stocker par le développeur pour chaque utilisateur vendeur
        - NbMbPerUserSellerOnMainchain : Quantité de données à stocker par le développeur sur la Blockchain principale pour chaque utilisateur vendeur
        - NbMbPerUserSellerOnLayerTwo : Quantité de données à stocker par le développeur sur la plateforme de Layer two pour chaque utilisateur vendeur
        - NbMbPerUserSellerOnStorage : Quantité de données à stocker par le développeur sur la solution de stockage choisie pour chaque utilisateur vendeur

Nous avons choisi de reprendre les mêmes DataProperties que nous avions définis en section 4.1.5, relatives aux coûts des transactions et du stockage de données de l'application. Cependant, les deux types d'utilisateurs ne vont pas utiliser l'application de la même manière, donc un dédoublement des propriétés utilisateur est réalisé pour tenir compte de ces différences.

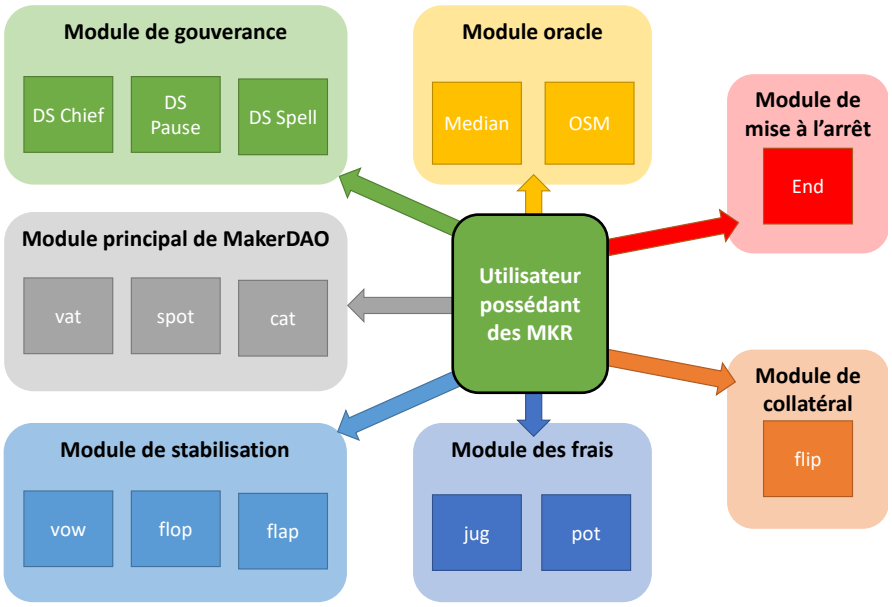

FIG. 4.7 : Exemple d'une DAO, adapté de `https://github.com/makerdao/dss/wiki#system-architecture`. Cette figure montre l'ensemble des services de MakerDAO qu'un utilisateur responsable de la gouvernance peut influer.

4.2.3 Les Organisations Autonomes Décentralisées

Les Organisations Autonomes Décentralisées, ou *Decentralized Autonomous Organization* (DAO) [158], sont des DApps fonctionnant sur le modèle des entreprises. Ainsi, à l'instar des actions d'une entreprise, une DAO possède un token, qui offre à leurs possesseurs un droit d'accès à la gouvernance de la DAO. Cette gouvernance peut concerner la mise à jour de l'application, de ses propriétés, etc.

Un exemple connu de DAO est MakerDAO, qui assure la gestion de la *stablecoin* Dai [43]. Le token associé à la DAO est Maker, et les possesseurs de Maker peuvent voter pour influer sur de nombreux paramètres de l'application. Les différentes interactions de gouvernance sur cette DAO sont présentées FIGURE 4.7. Nous voyons dans ce schéma l'ensemble des modules de la DApp que les possesseurs du token MKR peuvent influer. Ces influences peuvent être relatives au fonctionnement classique de l'application, telles qu'une modification des taux de prêts ou de la stabilisation du protocole, mais également sur les fonctionnements critiques tels que la mise à l'arrêt du système en cas d'événement particulier comme une faille ou un comportement anormal des marchés.

Finalement, notre ontologie propose de modéliser les DAO de manière similaire que les NFT Marketplace, en différenciant deux types d'utilisateurs :

- Ceux qui sont responsables de la gouvernance de la DAO,
- Ceux qui utilisent simplement les services de la DAO.

Les DataProperties spécifiques aux DAO sont les suivantes, pour lesquels nous reprenons la même nomenclature que précédemment :

- UserForGovernance : Propriétés relatives à un utilisateur assurant la gouvernance de la DAO
    - NbTxPerUserForGovernance : Nombre de transactions pour chaque utilisateur assurant la gouvernance
        - NbTxPerUserForGovernanceOnMainchain : Nombre de transactions sur la Blockchain principale pour chaque utilisateur assurant la gouvernance
        - NbTxPerUserForGovernanceOnLayerTwo : Nombre de transactions sur la plateforme de Layer two pour chaque utilisateur assurant la gouvernance
    - NbMbPerUserForGovernance : Quantité de données à stocker par le développeur pour chaque utilisateur
        - NbMbPerUserForGovernanceOnMainchain : Quantité de données à stocker par le développeur sur la Blockchain principale pour chaque utilisateur assurant la gouvernance
        - NbMbPerUserForGovernanceOnLayerTwo : Quantité de données à stocker par le développeur sur la plateforme de Layer two pour chaque utilisateur assurant la gouvernance
        - NbMbPerUserForGovernanceOnStorage : Quantité de données à stocker par le développeur sur la solution de stockage choisie pour chaque utilisateur assurant la gouvernance
- TargetUser : Propriétés relatives à un utilisateur cible de la DAO
    - NbTxPerTargetUser : Nombre de transactions pour chaque utilisateur cible
        - NbTxPerTargetUserOnMainchain : NbTxPerUserSellerOnMainchain : Nombre de transactions sur la Blockchain principale pour chaque utilisateur cible
        - NbTxPerTargetUserOnLayerTwo : Nombre de transactions sur la plateforme de Layer two pour chaque utilisateur cible
    - NbMbPerTargetUser : Quantité de données à stocker par le développeur pour chaque utilisateur cible
        - NbMbPerTargetUserOnMainchain : Quantité de données à stocker par le développeur sur la Blockchain principale pour chaque utilisateur cible

- NbMbPerTargetUserOnLayerTwo : Quantité de données à stocker par le développeur sur la plateforme de Layer two pour chaque utilisateur cible
- NbMbPerTargetUserOnStorage : Quantité de données à stocker par le développeur sur la solution de stockage choisie pour chaque utilisateur cible

4.2.4 Les oracles

Les oracles sont des solutions pour lire des données du monde extérieur au sein d'une Blockchain. Pour cela, les développeurs de DApp doivent définir des sources de données considérées comme sûres. Les valeurs prises par ces sources de données peuvent alors être utilisées en tant que source de vérité.

La DAO MakerDAO, par exemple, nécessite pour son fonctionnement que les smart contracts de la DAO aient accès aux valeurs actuelles du cours de différentes cryptomonnaies. En effet, les utilisateurs de MakerDAO peuvent récupérer des Dai en plaçant en collatéral différentes cryptomonnaies comme l'Ether. Le collatéral est une valeur monétaire, ici en cryptomonnaie, que l'utilisateur transfère au contrat. Il ne pourra récupérer cette monnaie que s'il rembourse le prêt en Dai octroyé par le contrat. Dans le cas où l'utilisateur ne rembourse pas son prêt, alors le contrat garde le collatéral en dédommagement. Le contrat a ainsi besoin de connaître la valeur de l'Ether pour savoir combien de Dai un utilisateur peut recevoir à partir de la valeur du collatéral qu'il a placé. Pour cela, une certaine liste de sources de données de cours de l'ETH est définie et gérée par Omnia Relay. Un système de consensus pour choisir la valeur finale prise en compte doit également être défini. Liu et. al [159] précisent l'aspect critique des informations reçues : si la coalition des acteurs autorisés à agir sur l'oracle agissent de façon malveillante, les monnaies mises en collatéral au sein du système deviennent vulnérables. Cet exemple est précisé dans la FIGURE 4.8.

Dans cette figure, plusieurs échanges de cryptomonnaies fournissent par leur API les cours de différentes cryptomonnaies en temps réel. Ensuite, Omnia agrège l'ensemble de ces valeurs, et les fournit aux contrats de MakerDAO. Les adresses d'Omnia sont sur liste blanche pour gérer l'accès à cette fonctionnalité des contrats. Les smart contracts calculent ensuite la valeur finale des cours qu'ils prennent en compte.

Au sein de notre ontologie, nous pouvons modéliser le fait qu'une certaine DApp utilise un Oracle. Cependant, cela n'a pas à priori d'impact sur les caractéristiques de la DApp que nous étudions, en termes de coût ou de latence.

4.2.5 Les chaînes logistiques

Le domaine des chaînes logistiques, ou *supply chain*, est un domaine où la traçabilité des transactions apportées par la Blockchain rend cette technologie

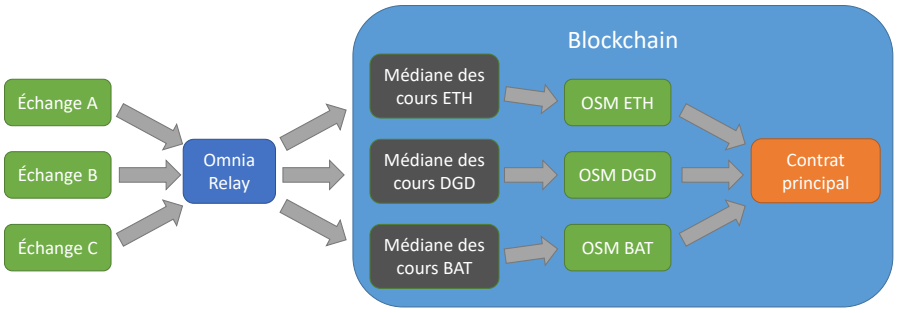

FIG. 4.8 : Exemple d'Oracle, inspiré de `https://docs.makerdao.com/smart-contract-modules/oracle-module`

pertinente pour ce domaine [160]. En particulier, il est intéressant d'avoir une traçabilité complète des produits, en fonction de leurs références. Chaque référence correspond à une famille de produits identiques. Par exemple, un vendeur de meubles peut proposer dans son catalogue deux références de tables et une références de chaise. Chacune de ces références peut alors être produit dans un certain nombre d'unité. Le vendeur pourrait avoir 2000 produits différents : 250 tables de chaque référence ainsi que 1500 chaises.

Ainsi, pour de telles DApps, une formalisation centrée sur les produits et les références produits est pertinente, et forme un Blockchain Pattern, présenté FIGURE 4.9. Cet exemple montre un processus d'élevage et de production de viande, dans lequel l'ensemble des acteurs impliqués dans ce processus est utilisateur de la DApp. Les différents acteurs scannent des QR-codes permettant la mise des données sur la Blockchain de façon simplifiée pour les utilisateurs. À la fin, un traçage de l'ensemble des produits peut être réalisé.

Les DataProperties spécifiques aux DApp de chaîne logistique sont les suivantes, pour lesquels nous reprenons la même nomenclature que précédemment :

- Product : Propriétés relatives à un produit spécifique de la chaîne logistique
  - NbTxPerProduct : Nombre de transactions pour chaque produit
    - NbTxPerProductOnMainchain : Nombre de transactions sur la Blockchain principale pour chaque produit
    - NbTxPerProductOnLayerTwo : Nombre de transactions sur la plateforme de Layer two pour chaque produit
  - NbMbPerProduct : Quantité de données à stocker pour chaque produit
    - NbMbPerProductOnMainchain : Quantité de données à stocker sur la Blockchain principale pour chaque produit

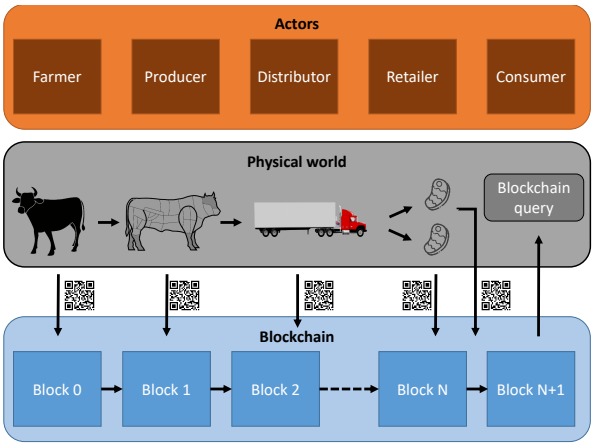

FIG. 4.9 : Exemple d'application de la Blockchain dans le domaine des chaînes logistiques, inspiré de `https://resolvesp.com/blockchains-supply-chains-part-ii`

    - NbMbPerProductOnLayerTwo : Quantité de données à stocker sur la plateforme de Layer two pour chaque produit
    - NbMbPerProductOnStorage : Quantité de données à stocker sur la solution de stockage choisie pour chaque produit

- Reference : Propriétés relatives à une référence produit spécifique de la chaîne logistique
    - NbTxPerReference : Nombre de transactions pour chaque référence
        - NbTxPerReferenceOnMainchain : Nombre de transactions sur la Blockchain principale pour chaque référence
        - NbTxPerReferenceOnLayerTwo : Nombre de transactions sur la plateforme de Layer two pour chaque référence
    - NbMbPerReference : Quantité de données à stocker pour chaque référence
        - NbMbPerReferenceOnMainchain : Quantité de données à stocker sur la Blockchain principale pour chaque référence
        - NbMbPerReferenceOnLayerTwo : Quantité de données à stocker sur la plateforme de Layer two pour chaque référence
        - NbMbPerReferenceOnStorage : Quantité de données à stocker sur la solution de stockage choisie pour chaque référence

Dans le cas des DApps de chaînes logistiques, nous reprenons le même schéma que celui défini en 4.1.5 et utilisé par exemple pour le Blockchain Pattern de NFT Markerplace. Cependant, ici nous ne différencions pas selon le type d'utilisateur, mais selon le type d'utilisation de l'application. Une DApp avec

une seule référence comportant de nombreux produits ne sera pas utilisée de la même façon qu'une DApp qui comporte de nombreuses références en quantité très limitée. En reprenant l'exemple des vendeurs de mobilier, la DApp d'un vendeur proposant 100000 produits à travers une unique référence n'aura pas les mêmes besoins qu'un vendeur proposant 50 produits à travers 25 références.

4.2.6 Les interfaces utilisateur non bloquantes

Un dernier Blockchain Pattern que nous avons formalisé correspond aux interfaces utilisateur non bloquantes, proposé par Ojha [161]. Avec ce type d'interfaces utilisateur, un utilisateur peut initier plusieurs actions, sans avoir à attendre la finalisation de la première action. Ce pattern est important dans le domaine de la Blockchain, puisque ce domaine repose beaucoup sur de la programmation événementielle. Ce pattern décrit ainsi qu'au sein d'une DApp, l'interface utilisateur ne doit pas être bloquante.

La FIGURE 4.10 montre un exemple d'utilisation de ce Blockchain Pattern.

Concrètement, pour notre ontologie, non avons choisis de différencier les actions qui peuvent être effectuées simultanément, et les actions qui doivent obligatoirement être séquentielles. Par exemple, pour une DApp permettant la création et la vente de NFT, un utilisateur qui souhaite en créer un doit charger un fichier image. Avant la fin du transfert de fichier, l'utilisateur garde le contrôle de l'application et peut modifier la description du NFT créé. Ces deux actions peuvent donc être effectuées simultanément. En revanche, lorsqu'un autre utilisateur souhaite acheter ce NFT, il doit forcément attendre la confirmation de la transaction avant de le posséder et de l'utiliser, par exemple dans un jeu. L'achat du NFT est une action bloquante pour son utilisation. Cependant, la transaction peut mettre plusieurs minutes avant d'être validée. Ainsi, l'application doit laisser le contrôle à l'utilisateur le temps que la transaction soit complétée. En effet, l'utilisateur peut décider d'acheter un autre NFT avant même que le premier ne soit acheté.

Les DataProperties spécifiques aux DApp implémentant le Blockchain Pattern d'interfaces utilisateur non bloquantes sont les suivantes :

- ActionQueueSize : Nombre d'actions totales réalisables
- ActionCriticalQueueSize : Nombre d'actions maximales à réaliser séquentiellement

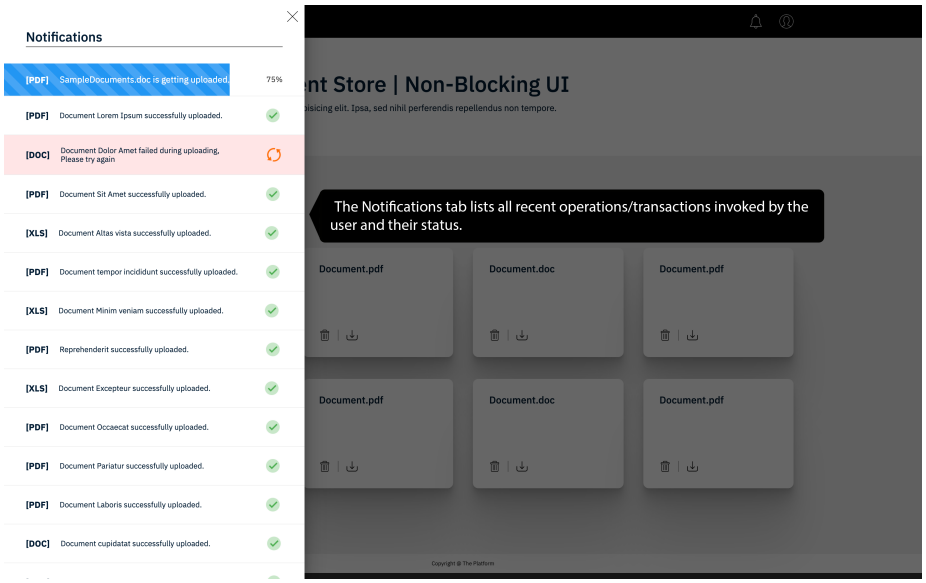

FIG. 4.10 : Exemple d'interface utilisateur non bloquante, repris de [161]

4.2.7 Synthèse sur les Blockchain Patterns

Pour chacun de ces design patterns Blockchain, nous avons défini des DataProperties utiles à leur modélisation. Ainsi, si l'on souhaite modéliser une application quelconque au sein de notre ontologie, il est possible d'utiliser les DataProperties afin de créer une modélisation adaptée. Nous n'avons pas eu besoin d'ajouter de nouveaux concepts ou de nouvelles propriétés d'objets. En effet, les concepts et les ObjectProperties existants suffisent à la formalisation des Blockchains Patterns présentés précédemment.

Il faudra ensuite écrire des règles SWRL qui permettent d'utiliser ces propriétés afin de déduire les caractéristiques de fonctionnement de l'application concrète. En effet, il est difficile d'écrire des règles SWRL génériques pour chaque pattern, qui n'ont pas besoin d'éléments spécifiques à l'application à modéliser. Cependant, l'étude des règles SWRL présentes au sein de l'ontologie permet aux chercheurs de facilement les adapter à leur application, et ainsi en déduire une modélisation exhaustive.

4.3 Synthèse

Au sein de ce chapitre, nous avons détaillé une contribution de recherche que nous avons réalisée. Nous avons présenté une ontologie générique du domaine de la Blockchain. Cette ontologie permet en particulier de formaliser une DApp telle que le demande notre méthodologie. Cette ontologie étend l'ontologie Blockchain existante EthOn. Cette approche permet entre autres d'assurer que toute mise à jour d'une ontologie est automatiquement répliquée dans les ontologies qui l'étendent. Notre ontologie propose aussi de nombreux éléments de modélisation, tels que des propriétés d'objets et des propriétés de données.

Les autres ontologies de la littérature ne permettent pas cette formalisation de DApps. Ainsi, notre contribution permet d'améliorer l'interopérabilité des DApps d'une part, et des services qui composent une DApp d'autre part.

En effet, l'ontologie proposée permet d'obtenir une représentation sémantique cohérente entre différentes DApps, ce qui améliore leur interopérabilité. Ainsi, deux DApps formalisées à partir de cette ontologie peuvent s'accorder sur une sémantique commune. Par exemple, un NFT Marketplace et une application de gestion de NFT ont besoin de posséder la même formalisation du concept de NFT, afin de s'assurer des bonnes interactions entre les deux applications.

D'autre part, cette ontologie propose une formalisation sémantique des services Blockchains existants, et de leur utilisation au sein d'une DApp. Cette formalisation est nécessaire à l'application de la première étape de notre méthodologie d'aide à la conception de DApps présentée au sein du chapitre 3.

L'utilisation de cette ontologie pour le développement de DApp, par le biais d'outils tels que des règles SWRL et des Blockchain Patterns, sera présentée au sein du prochain chapitre.

Chapitre 5

Utilisation de l'ontologie pour le développement d'applications Blockchain décentralisées interopérables

Afin d'utiliser l'ontologie du chapitre précédent pour modéliser des Applications Blockchain Décentralisées, nous devons écrire des règles SWRL, puis raisonner sur ces règles pour en déduire des axiomes. Cela permet de comparer, pour une application donnée, ses caractéristiques en cas d'utilisation d'un service Blockchain ou d'un autre.

5.1 Définition des axiomes SWRL

SWRL [162] est un langage permettant d'exprimer des règles agissantes sur une ontologie OWL. Chaque règle que nous écrivons peut servir à un moteur d'inférence afin de compléter l'ontologie existante. Une fois l'ontologie complétée, nous pouvons alors définir de nouvelles règles au sein d'un processus itératif.

Les règles SWRL suivent toutes le même schéma de fonctionnement. Nous définissons des axiomes qui contraignent des individus ou des relations entre individus. Ces axiomes peuvent faire partie de l'antécédent, ou de la conclusion de la règle. Si tous les axiomes de l'antécédent sont vérifiés, alors tous les axiomes de la conclusion le sont également.

5.1.1 Axiomes recherchés et règles obtenues

Nous avons défini une liste d'axiomes de base pour notre ontologie. Nous détaillons dans les paragraphes suivants les motivations de l'écriture de ces règles.

Tout d'abord, les règles 1, 1bis et 2 ont pour but de définir des contraintes de base sur les concepts de la technologie Blockchain, et permettent donc de s'assurer que notre ontologie formalise bien la technologie.

Les règles 3, 4, 5 et 6 définissent des contraintes sur les liens entre la Blockchain Ethereum et sa sidechain Matic. Elles permettent donc de comprendre comment ces deux Blockchains opèrent ensemble, ce qui améliore leur interopérabilité.

Enfin, les dix règles spécifiques, numérotées A à D4, définissent les contraintes liées aux DApp. Par exemple, nous déduisons les caractéristiques de coûts d'une telle application. Elles nous permettent donc d'obtenir des résultats concrets une fois une DApp modélisée au sein de notre ontologie.

Règle 1

Une transaction appartenant à la liste de transactions d'un bloc valide est valide.

```
1 gbo:Transaction( ?x)  ∧
2 gbo:TransactionList( ?list)  ∧
3 gbo:ValidBlock( ?b)  ∧
4 gbo:Contains( ?b,  ?list)  ∧
5 gbo:Contains( ?list,  ?x)
6 →
7 gbo:ValidTransaction( ?x)
```

Listing 5.1 : Règle 1

La règle du listing 5.1 est simple : les lignes 1 à 3 définissent les trois individus utilisés : une transaction `?x`, une liste de transaction `?list`, et un bloc valide `?b`. Ensuite, les lignes 4 et 5 définissent leurs relations d'inclusion : le bloc `?b` contient la liste `?list`, qui contient à son tour la transaction `?x`. Enfin, la ligne 7 définit le conséquent de la règle, qui est que la transaction `?x` est valide si tous les axiomes de l'antécédent sont valides.

Avec l'aide du Plugin *Aided OWL Notation* (AOWLN) [163] pour protégé, une représentation graphique est obtenue pour la première de ces règles, comme le montre la FIGURE 5.1. À gauche, l'antécédent de la règle. À droite, son conséquent.

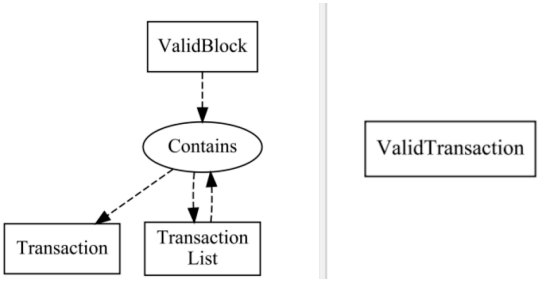

FIG. 5.1 : Représentation graphique de la règle 1. À gauche, enchaînement des propriétés de l'antécédent de la règle. À droite, le conséquent de la règle.

Règle 1bis

Un bloc possédant dans sa liste de transactions une transaction invalide n'est pas valide.

```
1 gbo:FailedTransaction( ?x)  ∧
2 gbo:TransactionList( ?list)  ∧
3 gbo:Block( ?b)  ∧
4 gbo:Contains( ?b,  ?list)  ∧
5 gbo:Contains( ?list,  ?x)
6 →
7 gbo:InvalidBlock( ?b)
```

Listing 5.2 : Règle 1bis

La règle du listing 5.2 est donc complémentaire à la première règle, et repose sur le même principe de construction. Les lignes 1 à 3 définissent les trois individus utilisés : une transaction échouée `?x`, une liste de transaction `?list`, et un bloc `?b`. Ensuite, les lignes 4 et 5 définissent leurs relations d'inclusion, puis la ligne 7 définit le conséquent de la règle.

De même que précédemment, nous utilisons le Plugin AOWLN pour obtenir une représentation graphique de cette règle en FIGURE 5.2.

2ème règle

Un bloc valide sur une Blockchain utilisant la PoW est nécessairement un bloc miné par une personne sur cette même Blockchain.

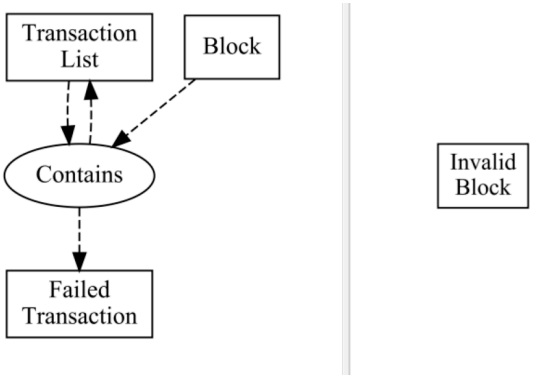

FIG. 5.2 : Représentation graphique de la règle 1bis. À gauche, enchaînement des propriétés de l'antécédent de la règle. À droite, le conséquent de la règle.

```
1 gbo:ValidBlock( ?b) ∧
2 gbo:Blockchain( ?blockchain) ∧
3 gbo:ProofOfWork( ?pow) ∧
4 gbo:BlockchainAccount( ?x) ∧
5 gbo:Uses( ?blockchain, ?pow) ∧
6 gbo:Contains( ?blockchain, ?b) ∧
7 gbo:HasBeneficiary( ?b, ?x)
8 →
9 gbo:MinesFor( ?x, ?blockchain)
```

Listing 5.3 : Règle 2

Comme les règles précédentes, cette deuxième règle, définit dans le listing 5.3, traite des concepts de base des Blockchains. Les quatre premières lignes définissent les individus utilisés : une Blockchain `?blockchain`, un bloc valide `?b`, la preuve de travail `?pow` et un compte Blockchain `?x`. La ligne 5 s'assure que la Blockchain considérée fonctionne bien par PoW, les autres méthodes de consensus ne possédant pas de mineurs. Les lignes 6 et 7 définissent respectivement les liens « la Blockchain `?blockchain` contient le bloc `?b` » et « le bloc `?b` est miné par le compte `?x` ». Enfin, la dernière ligne conclut que `?x` mine pour `?blockchain`.

3ème règle

Si une transaction Matic est référencée par son hash au sein d'une transaction Ethereum valide, alors elle est également valide.

```
gbo:MaticTransaction(?x) ∧
gbo:EthereumTransaction(?y) ∧
gbo:ValidTransaction(?y) ∧
gbo:HasHash(?x, ?h) ∧
gbo:Contains(?y, ?h)
→
gbo:ValidTransaction(?x)
```

Listing 5.4 : Règle 3

La règle du listing 5.4, ainsi que les trois suivantes, traite des liens entre la Blockchain Ethereum et sa sidechain Matic. Ici, nous considérons qu'une transaction Matic est entièrement validée par Ethereum via un Checkpoint.

Nous définissons lignes 1 à 3 une transaction Matic, une transaction Ethereum qui est également une transaction valide. Les lignes 4 et 5 énoncent que la transaction Ethereum doit contenir le hash de la transaction Matic. Dans ce cas, la ligne 7 énonce que cette transaction Matic est valide.

4ème règle

Si une transaction Matic existe depuis plus d'une semaine, alors elle est soit valide, soit invalide. Cela signifie que cette transaction n'est plus en attente de validation d'un *challenge*, puisque le délai pour lequel une personne peut apporter une preuve de non-validité de la transaction est écoulé.

```
gbo:Matic(?blockchain) ∧
gbo:MaticTransaction(?x) ∧
gbo:TxSubmittedTime(?x, ?y) ∧
gbo:Contains(?blockchain, ?x) ∧
gbo:LastBlockTime(?blockchain, ?z) ∧
swrlb:dayTimeDuration(?dayTimeDurationLiteralWeek, 7, 0,
  0, 0) ∧
swrlb:subtract(?dayTimeDurationVariable, ?z, ?y) ∧
swrlb:greaterThan(
    ?dayTimeDurationLiteralWeek,
    ?dayTimeDurationVariable)
→
gbo:SettledTransaction(?x)
```

Listing 5.5 : Règle 4

Le listing 5.5 définit une quatrième règle, un peu plus complexe que les précédentes. Elle repose sur le fait que Matic fonctionne grâce à un système de *Fraud proof* tel que défini dans la section 2.4.1. Ainsi, si une transaction entre Matic et Ethereum est propagée par un nœud, les autres nœuds du réseau ont

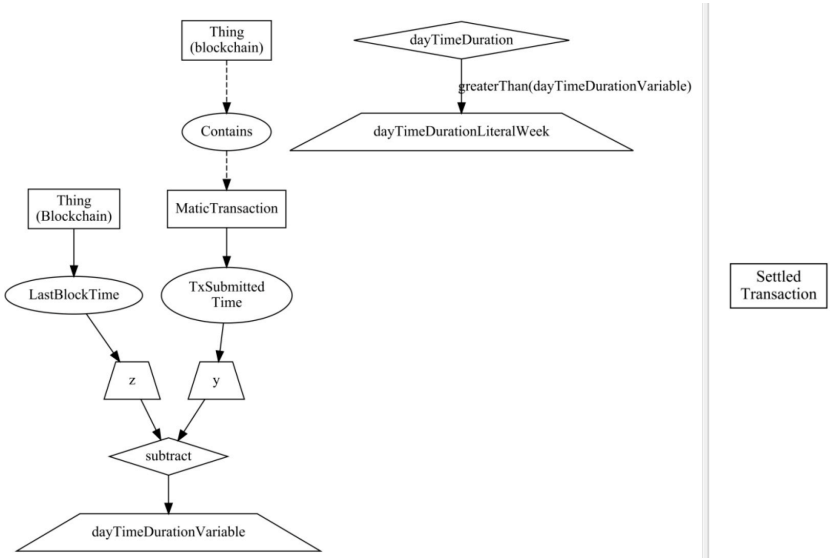

FIG. 5.3 : Représentation graphique de la règle 4. À gauche, enchaînement des propriétés de l'antécédent de la règle. À droite, le conséquent de la règle.

une semaine afin de contredire les affirmations données. C'est pourquoi, si une transaction Matic est plus ancienne que 7 jours, alors elle est soit validée, soit rejetée par le réseau.

Ici, les lignes 1 à 4 définissent une transaction Matic `?x` soumise à un temps `?y` et incluse dans la Blockchain Matic. Dans les lignes 5 à 10, nous utilisons des fonctions fournies par SWRL, appelées *built-ins*, qui permettent la gestion du temps. Nous calculons l'écart de temps entre le temps de soumission de la transaction `?x` et le temps du dernier bloc de Matic, `?z`. Nous comparons ensuite cet écart à un littéral de temps décrivant le délai d'une semaine. La ligne 12 indique que si l'écart de temps est supérieur à une semaine, alors la transaction est *settled*, c'est-à-dire soit valide soit invalide.

De nouveau, AOWLN nous permet d'obtenir la FIGURE 5.3. Nous voyons graphiquement que la règle fait intervenir des contraintes plus complexes que précédemment.

5ème règle

La sidechain Matic est périodiquement sauvée sur la mainchain Ethereum (sous la forme d'un *Merkel root* des nouveaux blocs).

```
1  gbo:Matic(?m) ∧
2  gbo:LastBlockTime(?m, ?y) ∧
3  gbo:LastCheckpointTime(?m, ?z) ∧
4  swrlb:dayTimeDuration(?dayTimeDurationLiteralHalfHour, 0,
       0, 30, 0) ∧
5  swrlb:subtract(?dayTimeDurationVariable, ?z, ?y) ∧
6  swrlb:greaterThan(
7       ?dayTimeDurationLiteralHalfHour,
8       ?dayTimeDurationVariable) ∧
9  gbo:Ethereum(?eth) ∧
10 gbo:NextBlock(?eth, ?b) ∧
11 gbo:MaticCheckpoint(?h)
12 →
13 gbo:Contains(?b, ?h)
```

Listing 5.6 : Règle 5

Semblablement à la règle 4, nous utilisons les *built-ins* de temps de SWRL. La règle du listing 5.6 propose d'inclure un Checkpoint Matic au prochain bloc miné sur Ethereum. Cependant, il faut être conscient que cette règle n'est pas obligatoirement respectée, mais reflète plutôt le fonctionnement attendu de Matic. En effet, chaque mineur choisit les transactions qu'il inclut dans ses blocs. Le mineur du prochain bloc peut ainsi choisir de ne pas inclure le Checkpoint Matic, même s'il a déjà été propagé à travers le réseau. En effet, les utilisateurs d'une Blockchain propagent des transactions souhaitées à travers le réseau : les transactions en attente de validation sont incluses dans le *mempool*. Cependant, rien n'empêche un mineur de ne pas inclure une certaine transaction du bloc créé.

Nous définissons lignes 1 à 3 la Blockchain Matic, la date de son dernier bloc et la date du dernier Checkpoint effectué. Ensuite, lignes 4 à 8, nous regardons si l'écart entre ces deux temps est supérieur à 30 minutes. Dans les lignes 9 à 11, nous définissons la Blockchain Ethereum, son prochain bloc, et un Checkpoint Matic. Enfin, la ligne 13 permet de conclure que si le dernier Checkpoint Matic date de plus de 30 minutes, alors un nouveau Checkpoint sera inclus dans le prochain bloc d'Ethereum.

6ème règle

Une transaction Matic est incluse dans un checkpoint Matic si celui-ci est soumis entre la date de soumission de la transaction Matic, met pas au delà de 30 minutes.

```
1  gbo:MaticTransaction(?tx) ∧
2  gbo:TxSubmittedTime(?tx, ?x) ∧
3  gbo:MaticCheckpoint(?checkpoint) ∧
4  gbo:TxSubmittedTime(?checkpoint, ?y) ∧
5 temporal:add(?xPlusHalfHour, ?x, 30, "Minutes") ∧
6 temporal:before(?x, ?y) ∧
7 temporal:before(?y, ?xPlusHalfHour)
8   →
9  gbo:IncludedIn(?x, ?checkpoint) ∧
10 gbo:Contains(?checkpoint, ?x)
```

Listing 5.7 : Règle 6

La règle du listing 5.7 permet de lier une transaction Matic à la transaction Ethereum dans lequel elle est inclut, par le processus de checkpoint. Les lignes 1 à 4 permettent de définir une transaction Matic `?tx` soumise à un temps `?x`, ainsi qu'un checkpoint Matic `?checkpoint` soumis à un temps `?y`. Ensuite, nous utilisons les *built-ins* de temps temporal [164] qui permettent certains calculs sur les dates, de façon plus robuste que ceux inclus dans SWRL. Les lignes 5 à 7 permettent de s'assurer que la transaction est plus ancienne que le checkpoint, mais pas de plus de 30 minutes. Si ces contraintes ne sont pas vérifiées, la transaction serait incluse au sein d'un autre checkpoint. Les lignes 9 et 10 permettent de définir les propriétés d'objets liant la transaction Matic et son checkpoint : la transaction `?tx` est incluse dans le checkpoint, et le checkpoint `?checkpoint` contient la transaction.

Règle spécifique A

Les règles spécifiques A à D4 permettent de contraindre le concept de DApp et leur propriétés, en fonction des services utilisés par l'application.

La règle du listing 5.8 permet de déduire les coûts développeurs d'une DApp, à partir du nombre de transactions à réaliser, de la quantité de données à stocker, et des technologies utilisées par l'application. Pour cette règle et pour l'ensemble des règles spécifiques aux DApps, nous réutilisons la terminologie *Decentralized Blockchain Applications* (DBA) pour formaliser les DApps.

```
1 sbo:DecentralizedBlockchainApplication(?dba) ∧
2 gbo:Uses(?dba, ?blockchain) ∧
3 gbo:Blockchain(?blockchain) ∧
4 gbo:Uses(?dba, ?l2) ∧
5 sbo:LayerTwo(?l2) ∧
6 gbo:Uses(?dba, ?storage) ∧
7 sbo:Storage(?storage) ∧
8 sbo:NbTxDevPerUserOnMainchain(?dba, ?a) ∧
```

```
sbo:CostTx( ?blockchain, ?b)  ∧
sbo:NbTxDevPerUserOnLayerTwo( ?dba,?c)  ∧
sbo:CostTx( ?l2, ?d)  ∧
sbo:NbMbDevPerUserOnMainchain( ?dba, ?e)  ∧
sbo:CostPerMb( ?blockchain, ?f)  ∧
sbo:NbMbDevPerUserOnLayerTwo( ?dba, ?g)  ∧
sbo:CostPerMb( ?l2, ?h)  ∧
sbo:NbMbDevPerUserOnStorage( ?dba, ?i)  ∧
sbo:CostPerMb( ?storage, ?j)  ∧
swrlb:multiply( ?product1, ?a, ?b)  ∧
swrlb:multiply( ?product2, ?c, ?d)  ∧
swrlb:multiply( ?product3, ?e, ?f)  ∧
swrlb:multiply( ?product4, ?g, ?h)  ∧
swrlb:multiply( ?product5, ?i, ?j)  ∧
swrlb:add( ?k, ?product1, ?product2, ?product3, ?product4,
    ?product5)
→
sbo:CostDevPerUser( ?dba, ?k)
```

Listing 5.8 : Règle spécifique A

Les règles A et B sont relativement complexes puisqu'elles consistent en le calcul des coûts associés à la DApp considérée. SWRL, et plus généralement OWL, ne sont pas particulièrement adaptés aux calculs.

Les lignes 1 à 7 définissent une DApp `?dba` et les technologies utilisées par celle-ci : la Blockchain utilisée `?blockchain`, la solution de scalabilité des transactions `?l2`, ainsi que la solution de stockage `?storage`. Si une DApp n'utilise pas un de ces services, l'individu sera simplement un `NullLayerTwo` ou un `NullStoage`, ce qui ne gêne pas dans la suite de la règle.

Ensuite, les lignes 8 à 17 visent à définir les caractéristiques de la DApp, donc le nombre de transactions à réaliser par le développeur et la quantité de données à stocker, ainsi que les propriétés des services Blockchain employés : les coûts des transactions et du stockage de données. Les lignes 18 à 23 utilisent ces données pour calculer le coût de la DApp, qui est appliqué en propriété de la DApp en ligne 25. La formule de calcul est la suivante :

$$\begin{aligned}\text{CostDevPerUser} &= \text{NbTxDevPerUserOnMainchain} \times \text{CostTx}_{\text{Blockchain}} \\ &+ \text{NbTxDevPerUserOnLayerTwo} \times \text{CostTx}_{\text{LayerTwo}} \\ &+ \text{NbMbDevPerUserOnMainchain} \times \text{CostPerMb}_{\text{Blockchain}} \\ &+ \text{NbMbDevPerUserOnLayerTwo} \times \text{CostPerMb}_{\text{LayerTwo}} \\ &+ \text{NbMbDevPerUserOnStorage} \times \text{CostPerMb}_{\text{Storage}}\end{aligned}$$

Règle spécifique B

Le listing 5.9 décrit cette règle spécifique B, qui permet de déduire les coûts utilisateur d'une DApp, à partir du nombre de transactions à réaliser, de la quantité de données à stocker, et des technologies utilisées par l'application. Cette règle est donc analogue à la règle A, mais pour des coûts utilisateur et non-développeurs.

```
1  sbo:DecentralizedBlockchainApplication(?dba) ∧
2  gbo:Uses(?dba, ?blockchain) ∧
3  gbo:Blockchain(?blockchain) ∧
4  gbo:Uses(?dba, ?l2) ∧
5  sbo:LayerTwo(?l2) ∧
6  gbo:Uses(?dba, ?storage) ∧
7  sbo:Storage(?storage) ∧
8  sbo:NbTxPerUserOnMainchain(?dba,?a) ∧
9  sbo:CostTx(?blockchain, ?b) ∧
10 sbo:NbTxPerUserOnLayerTwo(?dba,?c) ∧
11 sbo:CostTx(?l2, ?d) ∧
12 sbo:NbMbPerUserOnMainchain(?dba, ?e) ∧
13 sbo:CostPerMb(?blockchain, ?f) ∧
14 sbo:NbMbPerUserOnLayerTwo(?dba, ?g) ∧
15 sbo:CostPerMb(?l2, ?h) ∧
16 sbo:NbMbPerUserOnStorage(?dba, ?i) ∧
17 sbo:CostPerMb(?storage, ?j) ∧
18 swrlb:multiply(?product1, ?a, ?b) ∧
19 swrlb:multiply(?product2, ?c, ?d) ∧
20 swrlb:multiply(?product3, ?e, ?f) ∧
21 swrlb:multiply(?product4, ?g, ?h) ∧
22 swrlb:multiply(?product5, ?i, ?j) ∧
23 swrlb:add(?k, ?product1, ?product2, ?product3, ?product4,
       ?product5)
24 →
25 sbo:CostUser(?dba, ?k)
```

Listing 5.9 : Règle spécifique B

Cette règle reproduit exactement le même schéma que la précédente, mais considère les coûts utilisateur au lieu de développeurs. Les lignes 1 à 7 définissent une DApp `?dba` et les technologies utilisées par celle-ci. Ensuite, les lignes 8 à 17 visent à définir les caractéristiques de la DApp. Les lignes 18 à 23 utilisent ces données pour calculer le coût de la DApp, qui est appliqué en propriété de la DApp en ligne 25. La formule de calcul est la suivante :

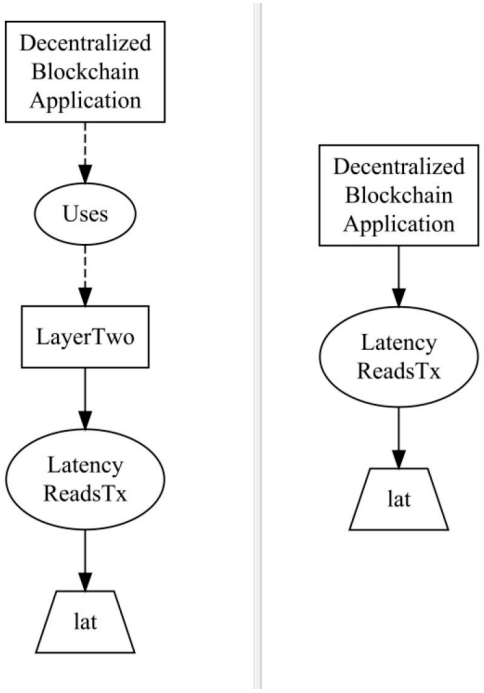

FIG. 5.4 : Représentation graphique de la règle spécifique C1. À gauche, enchaînement des propriétés de l'antécédent de la règle. À droite, le conséquent de la règle.

$$\begin{aligned}\text{CostPerUser} &= \text{NbTxPerUserOnMainchain} \times \text{CostTx}_{\text{Blockchain}} \\ &+ \text{NbTxPerUserOnLayerTwo} \times \text{CostTx}_{\text{LayerTwo}} \\ &+ \text{NbMbPerUserOnMainchain} \times \text{CostPerMb}_{\text{Blockchain}} \\ &+ \text{NbMbPerUserOnLayerTwo} \times \text{CostPerMb}_{\text{LayerTwo}} \\ &+ \text{NbMbPerUserOnStorage} \times \text{CostPerMb}_{\text{Storage}}\end{aligned}$$

Règles spécifiques C

Les règles décrites ci-dessous permettent de déduire les latences (en lecture et en écriture) des transactions d'une DApp.

La FIGURE 5.4 montre le graphique obtenu avec AOWLN pour la première de ces règles.

RuleSpec C1 : Cas général en lecture

```
sbo:DecentralizedBlockchainApplication( ?dba)  ∧
gbo:Uses( ?dba,  ?l2)  ∧
sbo:LayerTwo( ?l2)  ∧
sbo:LatencyReadsTx( ?l2,  ?lat)
→
sbo:LatencyReadsTx( ?dba,  ?lat)
```

Listing 5.10 : Règle spécifique C1

La règle du listing 5.10 permet de déduire les latences en lecture des transactions d'une DApp, lorsqu'un service de scalabilité des transactions est utilisé par l'application.

RuleSpec C2 : Cas d'un NullLayerTwo en lecture

```
sbo:DecentralizedBlockchainApplication( ?dba)  ∧
gbo:Uses( ?dba,  ?l2)  ∧
sbo:NullLayerTwo( ?l2)  ∧
gbo:Uses( ?dba,  ?blockchain)  ∧
gbo:Blockchain( ?blockchain)  ∧
sbo:LatencyReadsTx( ?blockchain,  ?lat)
→
sbo:LatencyReadsTx( ?dba,  ?lat)
```

Listing 5.11 : Règle spécifique C2

La règle du listing 5.11 a le même objectif que la précédente. Ici, nous gérons à part le fait qu'une DApp peut ne pas implémenter de service de scalabilité. La classe `sbo:NullLayerTwo` permet de gérer ce cas facilement.

RuleSpec C3 : Cas général en écriture

```
sbo:DecentralizedBlockchainApplication( ?dba)  ∧
gbo:Uses( ?dba,  ?l2)  ∧
sbo:LayerTwo( ?l2)  ∧
sbo:LatencyWritesTx( ?l2,  ?lat)
→
sbo:LatencyWritesTx( ?dba,  ?lat)
```

Listing 5.12 : Règle spécifique C3

La règle C3, définit par le listing 5.12 ; permet de déduire les latences en écriture des transactions d'une DApp, lorsqu'un service de scalabilité des transactions est utilisé par l'application.

RuleSpec C4 : Cas d'un NullLayerTwo en écriture

```
sbo:DecentralizedBlockchainApplication(?dba) ∧
gbo:Uses(?dba, ?l2) ∧
sbo:NullLayerTwo(?l2) ∧
gbo:Uses(?dba, ?blockchain) ∧
gbo:Blockchain(?blockchain) ∧
sbo:LatencyWritesTx(?blockchain, ?lat)
→
sbo:LatencyWritesTx(?dba, ?lat)
```

Listing 5.13 : Règle spécifique C4

La règle C4 proposée dans le listing 5.13 a le même objectif que la précédente. Ici, nous gérons à part le fait qu'une DApp peut ne pas implémenter de service de scalabilité. La classe `sbo:NullLayerTwo` permet de gérer ce cas facilement.

Règles spécifiques D

Les règles décrites ci-dessous permettent de déduire les latences (en lecture et en écriture) du stockage des données d'une DApp.

RuleSpec D1 : Cas général en lecture

```
sbo:DecentralizedBlockchainApplication(?dba) ∧
gbo:Uses(?dba, ?storage) ∧
sbo:Storage(?storage) ∧
sbo:LatencyReadsStorage(?storage, ?lat)
→
sbo:LatencyReadsStorage(?dba, ?lat)
```

Listing 5.14 : Règle spécifique D1

La règle du listing 5.14 permet de déduire les latences en lecture du stockage de données d'une DApp, lorsqu'un service de stockage est utilisé par l'application.

RuleSpec D2 : Cas d'un NullStorage en lecture

```
sbo:DecentralizedBlockchainApplication(?dba) ∧
gbo:Uses(?dba, ?storage) ∧
sbo:NullStorage(?storage) ∧
gbo:Uses(?dba, ?blockchain) ∧
gbo:Blockchain(?blockchain) ∧
sbo:LatencyReadsStorage(?blockchain, ?lat)
→
sbo:LatencyReadsStorage(?dba, ?lat)
```

Listing 5.15 : Règle spécifique D2

La règle du listing 5.15 a le même objectif que la précédente. Ici, nous gérons à part le fait qu'une DApp peut ne pas implémenter de service de stockage. La classe `sbo:NullStorage` permet de gérer ce cas facilement.

RuleSpec D3 : Cas général en écriture

```
sbo:DecentralizedBlockchainApplication(?dba) ∧
gbo:Uses(?dba, ?storage) ∧
sbo:Storage(?storage) ∧
sbo:LatencyWritesStorage(?storage, ?lat)
→
sbo:LatencyWritesStorage(?dba, ?lat)
```

Listing 5.16 : Règle spécifique D3

Le listing 5.16 présente une nouvelle règle, qui permet de déduire les latences en écriture du stockage de données d'une DApp, lorsqu'un service de stockage est utilisé par l'application.

RuleSpec D4 : Cas d'un NullStorage en écriture

```
sbo:DecentralizedBlockchainApplication(?dba) ∧
gbo:Uses(?dba, ?storage) ∧
sbo:NullStorage(?storage) ∧
gbo:Uses(?dba, ?blockchain) ∧
gbo:Blockchain(?blockchain) ∧
sbo:LatencyWritesStorage(?blockchain, ?lat)
→
sbo:LatencyWritesStorage(?dba, ?lat)
```

Listing 5.17 : Règle spécifique D4

La règle du listing 5.17 a le même objectif que la précédente. Ici, nous gérons à part le fait qu'une DApp peut ne pas implémenter de service de stockage. La classe `sbo:NullStorage` permet de gérer ce cas facilement.

5.1.2 Problématiques de l'Open World Assumption

OWL est un langage supportant l'*Open-World Assumption* (OWA). L'OWA signifie que OWL considère comme potentiellement vrai tout ce qui n'a pas été défini dans l'ontologie. Cet aspect en particulier limite considérablement les inférences que l'on peut faire sur notre cas d'application. En effet, la Blockchain étant une technologie décentralisée et distribuée, il est impossible de savoir qu'une certaine donnée n'existe pas. Concrètement, pour la règle 5 que nous avons définie, nous avons dû définir le concept de CheckpointMatic, mais nous ne pouvons pas récupérer l'ensemble des blocs Matic sur une période de temps.

5.1.3 Inductions trouvées

Au sein de la section 4.1.4, nous avons défini trois individus sbo:_DBA1, sbo:_DBA2 et sbo:_DBA3. Pour chacun de ces individus, nous avons déterminé certaines valeurs de DataProperties après modélisation des applications. Nous avons également déterminé des valeurs usuelles de coûts de transaction et de stockage pour les technologies utilisées.

Grâce à ces informations ainsi que les règles décrites précédemment, nous avons pu utiliser le moteur d'inférences Drools [165] afin de déduire de nouveaux axiomes pour notre ontologie. Nous avons choisi ce raisonneur puisqu'il est directement intégré avec le plugin Protégé SWRLTab [166] utilisé pour nos règles SWRL. Le raisonneur doit dans un premier temps traduire l'ontologie OWL et les règles SWRL en Drools, puis exécuter le moteur d'inférence, puis traduit enfin le résultat en axiomes OWL, qui sont alors ajoutés automatiquement à l'ontologie. L'ensemble de ces trois tâches est exécuté en un temps négligeable de 750ms, et produit 469 nouveaux axiomes, dont 261 axiomes relatifs aux propriétés de données ou d'objets. Des exemples d'axiomes inférés sont présentés dans le listing 5.18. Ce premier axiome définit la latence en écriture de l'individu sbo:_DBA1. Le deuxième axiome définit que le checkpoint sbo:_MaticCheckpoint2 contient la transaction sbo:_MaticTransaction2. Nous détaillons ce deuxième axiome au sein de la section 5.2. L'ensemble des axiomes inférés relatifs aux propriétés de données ou d'objets se trouve dans l'annexe A. Nous omettons les autres axiomes, relatifs de l'égalité, des classes, ou des types de données par souci de lisibilité.

```
1 sbo:_DBA1 sbo :LatencyWrites "14.0"^^xsd:double
2 sbo:_MaticCheckpoint2 gbo :Contains sbo:_MaticTransaction2
```

Listing 5.18 : Inductions trouvées à partir des règles SWRL spécifiques aux DApp

Les inductions trouvées par le moteur d'inférences sont détaillées dans le listing 5.19 ainsi que dans le tableau 5.1

Pour les deux premiers cas d'application, les coûts développeurs sont les mêmes, 2€ par utilisateur d'après nos hypothèses. Cependant, les coûts utilisateur sont plus élevés pour le premier cas d'application. En effet, les utilisateurs réalisent des transactions sur Ethereum dans ce premier cas, alors qu'ils les réalisent sur Matic pour le deuxième. Les coûts utilisateurs sont donc de 10€ pour le premier cas, et de seulement 0.01€ pour le deuxième cas.

Le troisième cas d'application utilise également Matic, donc les coûts utilisateur sont les mêmes que pour la deuxième DApp. Cependant, l'utilisation de Matic en tant que Blockchain et non seulement comme une sidechain d'Ethereum permet également de réduire les coûts développeurs, qui passent de 2€ par utilisateur à seulement 0.002€.

Les résultats obtenus sont analysés dans la section suivante.

```
sbo:CostDevPerUser(_dba1, 2.0)
sbo:CostUser(_dba1, 10)
sbo:CostDevPerUser(_dba2, 2.0)
sbo:CostUser(_dba2, 0.01)
sbo:CostDevPerUser(_dba3, 0.002)
sbo:CostUser(_dba3, 0.01)
```

Listing 5.19 : Inductions trouvées à partir des règles SWRL spécifiques aux DApp

DApp	CostDevPerUser	CostUser
DBA1	2	10
DBA2	2	0.01
DBA3	0.002	0.01

Tab. 5.1 : Caractéristiques des 3 DApp définies trouvées par Drools, en euro moyen.

5.1.4 Analyse des résultats

Dans un premier temps, nous voyons que notre ontologie nous permet bien de différencier les caractéristiques des trois DApp modélisées. À partir de ces résultats obtenus, il nous est possible de mieux comprendre l'impact que les différentes technologies potentielles ont sur les propriétés de l'application.

Nous voyons que les différences de coûts, à la fois pour les développeurs et les utilisateurs de la DApp, ont une variance très importante selon la technologie utilisée pour développer l'application. Ici, nous voyons que :

- Les coûts d'une solution basée uniquement sur Ethereum sont considérables, autant pour les développeurs que les utilisateurs.

- L'utilisation de Matic en conjonction d'Ethereum réduit considérablement les coûts utilisateur. Les coûts développeurs ne sont pas réduit, puisqu'ils réalisent l'ensemble de leurs transactions sur Ethereum. Cette solution convient bien si la DApp ne comporte pas trop d'utilisateurs.
- L'utilisation seule de Matic ne change pas les coûts utilisateur, par rapport au deuxième cas considéré. Cependant, les coûts développeurs sont fortement réduits.

Une des limitations des résultats obtenus est la non-prise en compte des contraintes liées à la sécurité de la DApp. En effet, nous pourrions penser qu'il faut alors choisir d'utiliser uniquement Matic, en tant que Blockchain principale, pour ce cas d'usage, puisque les coûts sont réduits. Cependant, Matic utilise une méthode de consensus, la DPoS, qui possède une sécurité intrinsèque plus faible que la PoW d'Ethereum. Lorsque Matic est utilisé en tant que sidechain d'Ethereum, cet inconvénient devient moindre, puisqu'Ethereum assure une partie de la sécurité de la DApp. Ainsi, à cause de cette limitation, nous ne recommanderions pas l'utilisation seule de Matic. Si les coûts développeurs sont prohibitifs, nous conseillerons l'utilisation d'autres solutions de scalabilité des transactions développeurs, telles que les *rollups*.

5.2 Mise en correspondance entre deux plateformes Blockchains

Une application des règles SWRL écrites est la mise en correspondance entre deux plateformes Blockchains, dans notre cas Ethereum et Matic.

L'ontologie présentée en chapitre 4 permet, entre autres, de formaliser différents services Blockchains, dont des instances de Blockchains spécifiques. Au sein de cette ontologie, nous avons pu modéliser les concepts liés à Ethereum et ceux liés à Matic.

Une fois cette formalisation initiale terminée, nous avons dans ce chapitre contraint les concepts de l'ontologie par des règles SWRL. En particulier, les règles Règle 3 à Règle 6 permettent de contraindre les concepts relatifs à Ethereum, Matic, et leurs liens. En effet, la règle 5 indique que toutes les 30 minutes, un nouveau checkpoint Matic est créé sur Ethereum, par le biais d'une transaction Ethereum. La règle 3 permet d'indiquer que les transactions Matic qui sont incluses dans ce checkpoint sont validées, et la Règle 6 de lier les transactions Matic à leur checkpoint. Enfin, la règle 4 décrit le fonctionnement de la période de challenge d'une semaine liant Matic et Ethereum.

L'objectif est ensuite de faire fonctionner le raisonneur Drools sur l'ontologie et les règles SWRL définies, afin d'obtenir des éléments de comparaison entre

Matic et Ethereum. En effet, comprendre les différences entre les propriétés de, par exemple, une transaction Matic et une transaction Ethereum permet à un développeur de DApps de connaître les conséquences d'un changement de plateforme pour son application.

Comme indiqué précédemment, le moteur d'inférences à permis de définir certains axiomes liants Ethereum à Matic. En particulier, la Règle 6, permettant de faire correspondre une transaction Matic avec le checkpoint qui l'inclut, permet de trouver les inférences détaillées dans le listing 5.20. Ainsi, le checkpoint _MaticCheckpoint1 contient la première transaction, et le checkpoint _MaticCheckpoint2 contient les deux autres transactions.

Il s'agit bien du résultat attendu, puisque _MaticCheckpoint1, soumis à Ethereum le 18/10/2021 à 18h30, valide les transactions Matic soumissent ce même jour entre 18h00 et 18h30, ce qui est le cas pour _MaticTransaction1. De même, _MaticCheckpoint2 valide les transactions Matic soumissent entre 18h30 et 19h00, ce qui est le cas pour _MaticTransaction2 et _MaticTransaction3.

```
gbo:Contains(_MaticCheckpoint1, _MaticTransaction1)
gbo:IncludedIn(_MaticTransaction1, _MaticCheckpoint1)
gbo:Contains(_MaticCheckpoint2, _MaticTransaction2)
gbo:IncludedIn(_MaticTransaction2, _MaticCheckpoint2)
gbo:Contains(_MaticCheckpoint2, _MaticTransaction3)
gbo:IncludedIn(_MaticTransaction3, _MaticCheckpoint2)
```

Listing 5.20 : Inductions trouvées à partir de la règle 6

Ainsi, nous avons trouvé des inférences permettant de lier les concepts d'Ethereum et de Matic, ce qui améliore l'interopérabilité entre ces deux Blockchains.

Au sein de la section 6.2.4, dans le chapitre d'application de nos recherches, nous détaillerons l'utilisation concrète de cette mise en correspondance.

5.3 Synthèse

Dans ce chapitre, nous étudions l'utilisation de règles SWRL pour améliorer l'expressivité de l'ontologie décrite au sein du chapitre 4. En effet, SWRL permet d'obtenir des contraintes entre les concepts de l'ontologie plus simplement et efficacement qu'en utilisant uniquement OWL. Par exemple, lors de l'exécution des règles SWRL définies précédemment, 469 axiomes supplémentaires sont ajoutés à l'ontologie, qui auraient été difficiles à déterminer.

Les règles SWRL ont trois objectifs :

- Formaliser des contraintes fondamentales liées aux Blockchains, telles que les règles de validité des blocs et des transactions d'une Blockchain

- Formaliser les liens entre la Blockchain Ethereum et sa sidechain Matic
- Formaliser les propriétés des DApps, et en particulier les coûts et latences de ces applications

D'autres avantages des règles SWRL définies sont leur évolutivité et modularité. En effet, il est facile pour d'autres chercheurs ou développeurs de formaliser leurs applications en adaptant ces règles ou en en ajoutant de nouvelles. Par exemple, des règles analogues aux règles spécifiques pour DApps A et B peuvent être écrites pour formaliser les contraintes de coûts des DApps qui utilisent des Blockchain Patterns telles que décrites dans la section 4.2.

Enfin, SWRL possède d'autres applications dans le domaine des Blockchains. Choudhury et al. [167] propose en effet l'utilisation de règles SWRL afin de générer automatiquement des smart contracts à partir de documents détaillants les régulations d'un domaine quelconque. Ce genre d'application permet également d'aider au développement des DApps, et nos règles permettent d'améliorer leur formalisation.

Mise sous contrainte des concepts Blockchains

Lorsqu'un nouveau concept est formalisé au sein de notre ontologie, il nous a fallu le contraindre. Pour cela, une première étape est de considérer les relations avec les autres classes définies. Avec OWL, nous pouvons lier deux classes par des ObjectProperties. Dans notre cas, les Blockchains pouvant être vues comme des structures de données particulières, plusieurs de ses relations sont des relations d'inclusion ou de contenance.

Si nous souhaitons contraindre les données associées à une classe qui n'a pas de lien direct avec une autre classe existante, nous pouvons utiliser les DataProperties. Ces propriétés permettent de déclarer certaines propriétés associées à une classe.

Parfois, le langage OWL n'est pas assez expressif pour contraindre suffisamment des concepts. Dans ce cas, nous utilisons les règles SWRL pour contraindre plusieurs concepts, les ObjectProperties, et les DataProperties qui les lient.

Chapitre 6

Application à l'industrie du jeu vidéo

Nous appliquons notre méthodologie et notre ontologie au domaine du jeu vidéo. Pour simplifier, nous commençons par la présentation d'un cas simple : un jeu d'échecs Blockchain. Ensuite, nous présentons le cas réel du jeu Light Trail Rush, développé par B2Expand.

6.1 Contexte d'application de notre recherche

Cette section, nous présentons dans un premier temps le contexte industriel de notre étude, puis nous la placerons dans le contexte de l'écosystème Blockchain.

6.1.1 Contexte industriel de l'étude

Challenges de l'industrie du jeu vidéo Blockchain

Comme nous l'avions vu au sein de la section 1.5, le domaine du jeu vidéo Blockchain est particulier, et cela à un impact pour établir une interopérabilité forte de ce type d'applications. En effet, et malgré d'importantes avancées, l'industrie du jeu vidéo Blockchain doit encore faire face à des limitations, indépendamment de celles de la technologie Blockchain.

Tout d'abord, relatifs à la décentralisation. Il s'agit d'un important atout, comme énoncé précédemment, puisqu'elle permet de redonner le contrôle aux utilisateurs. Cependant, dans certains cas particuliers, ce manque de contrôle de la part des développeurs d'un jeu peut être problématique. Par exemple, il

peut être nécessaire de limiter le partage non censurable de contenu illégal ou non adapté au public visé. Des solutions techniques peuvent aider à contourner le problème, mais les solutions actuelles demandent des compromis, en termes de liberté utilisateur ou de complexité d'implémentation [168].

Ensuite, d'un point de vue technique, le manque de maturité des Blockchains et de leur interopérabilité implique pour les développeurs d'être liés à des écosystèmes spécifiques à une Blockchain donnée.

Enfin, même si la majorité du public cible des jeux vidéo à des connaissances techniques adéquates en informatique, les concepts liés à la sécurité cryptographique des portefeuilles Blockchains restent trop complexes pour toucher un grand nombre de joueurs. Les jeux actuels se restreignent donc à un public informé dans le domaine de la Blockchain et des cryptomonnaies.

D'autres développeurs choisissent de prendre en charge les clés des comptes Blockchains utilisateur afin d'éviter ce problème, mais cela réduit grandement l'intérêt d'intégrer la technologie Blockchain dans leurs jeux.

Blockchain Game Alliance

La Blockchain Game Alliance [169] est un consortium cofondé par B2Expand et 8 autres entreprises des domaines du jeu vidéo et de la Blockchain. Il vise à promouvoir l'utilisation de la Blockchain au sein des jeux vidéo, élaborer des standards pour l'industrie et faciliter le partage des recherches et avancées des membres.

Un groupe de travail a été formé au sein de l'alliance afin de proposer un standard pour la gestion des métadonnées des objets Blockchains au sein des jeux vidéo. Ce groupe de travail pourrait ainsi nous aider à améliorer l'interopérabilité de services Blockchains dans le domaine des jeux vidéo.

6.1.2 Contexte au sein de l'écosystème Blockchain

L'écosystème Blockchain comprend trois principaux types d'organisations. Certaines cherchent à développer les protocoles des Blockchains existantes ou à en développer de nouvelles. D'autres conçoivent des outils permettant une meilleure utilisation des Blockchains. Par exemple, c'est le cas de Metamask pour l'accès à des comptes Blockchains au sein d'un navigateur, ou d'OpenZeppelin pour des smart contracts de référence pour le développement sur Ethereum.

B2Expand se situe dans la dernière catégorie : il s'agit d'une entreprise qui cherche à tirer parti des propriétés des Blockchains au sein de ses applications. De nombreuses entreprises cherchent à développer des DApps aux objectifs divers. Cependant, la création et gestion de NFT, comme le propose B2Expand pour son jeu LTR, est un cas d'utilisation dominant dans l'écosystème Blockchain.

6.2 Travail réalisé

Nous présentons dans cette section le travail industriel réalisé qui permet d'appliquer et valider nos contributions de recherche décrites dans les chapitres précédents.

6.2.1 Modélisation

Différents jeux vidéo peuvent avoir de nombreuses caractéristiques différentes. Leur manière d'être jouée peut également varier considérablement selon le jeu choisi. Nous devons ainsi commencer notre l'application de nos recherches au domaine du jeu vidéo par une modélisation générique de jeux. Pour cela, nous pouvons considérer ce qui peut différer selon les jeux, ainsi que les caractéristiques communes à la grande majorité des jeux.

Pour ce premier point, nous devons établir des classifications de jeux, telles que [170]. Nous pouvons pour cela choisir différents axes, selon le nombre de joueurs d'une partie, la temporalité, c'est-à-dire si le jeu est en tour par tour ou en temps réel, ou encore le type de jeu, tel que jeu de stratégie, de First-Person Shooter (FPS), ou de Role Playing Game (RPG).

À priori, la Blockchain pourrait être intégrée à n'importe quel type de jeux. Cependant, certains jeux peuvent davantage se prêter à cette intégration. Par exemple, un jeu en temps réel implique des contraintes importantes sur la latence de jeu, ce qui contraint considérablement l'intégration de la technologie Blockchain.

Un des points que l'ont peut retrouver au cœur de nombreux jeux est le concept d'action d'un joueur. Un joueur peut, à un temps donné, effectuer un certain nombre d'actions qui vont influer sur l'état d'une partie. Ces actions possibles sont restreintes par les règles du jeu. Ainsi, un joueur d'échecs ne pourra pas déplacer son roi de plus d'une case.

La grande majorité des jeux possèdent ce qu'on appelle des *conditions de victoire*, qui définissent les états du jeu pour lequel on considère la partie terminée et le joueur a atteint son objectif. Des contre-exemples sont donnés par les jeux d'exploration, où un joueur cherche à explorer de lui-même le monde du jeu selon ses envies, mais aucune condition de victoire n'est définie objectivement.

6.2.2 Cas d'étude : le jeu d'échecs

Nous commençons les applications de notre méthodologie par un jeu connu par le grand public ainsi que par la recherche en général. Cela nous permet de baser notre travail sur des recherches antérieures, ainsi que de mieux pouvoir analyser les résultats obtenus par notre méthodologie. La modélisation de cette application de notre méthodologie a été étudiée dans notre publication [69].

FIG. 6.1 : Interface du jeu d'échecs sur `https://chess.com` . À gauche, le plateau de jeu, et à droite l'analyse de l'historique des coups joués.

Des motivations potentielles pour intégrer la technologie Blockchain au sein du jeu d'échecs sont :

- de permettre facilement à n'importe quel utilisateur de la Blockchain de parier sur les issues d'un match ou d'une compétition,
- sécuriser le classement Elo des joueurs sur un registre de compte transparent et immuable, et de parier facilement sur leur évolution,
- et, outre mesure, de tester la technologie sur un jeu connu ayant des caractéristiques particulières, les échecs étant un jeu au tour par tour avec deux joueurs.

Une des limitations de l'intégration de la Blockchain au sein de ce jeu est la possibilité de triche via l'utilisation de moteurs d'échecs tels que Komodo ou StockFish. La triche est commune dans l'écosystème des échecs en ligne. Cependant, la détection de triche aux échecs est un domaine actif de recherche [171], d'autant plus avec la multiplication des compétitions en ligne en 2020.

Modélisation d'un jeu d'échec Blockchain

La FIGURE 6.1 montre l'interface de jeu du site Chess.com. Cette interface montre le plateau de jeu en analyse, avec l'historique des coups joués lors de la partie.

Afin de définir les contraintes de cette DApp, nous devons en premier lieu définir une ontologie qui définit les différents concepts que notre jeu intègre. Nous étendons ainsi l'ontologie décrite dans le chapitre 3 afin d'intégrer les concepts liés aux échecs. Nous devons par exemple formaliser :

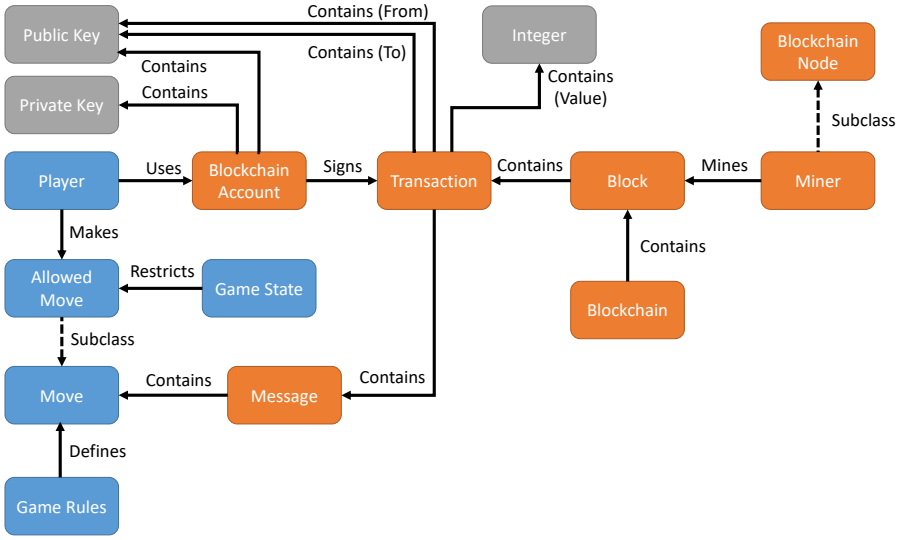

FIG. 6.2 : Une ontologie pour un jeu d'échecs Blockchain, publiée dans notre article [69].

- Les joueurs, qui sont les utilisateurs de l'application et disposent d'un compte Blockchain.
- Les actions qu'un joueur peut réaliser lors d'une partie, qui sont incluses dans des messages. Les échecs possèdent déjà une notation standardisée permettant de décrire ces actions. Par exemple, la notation « Be5 » signifie « Déplacer un fou sur la case e5 ».
- Les messages décrits dans le point précédent sont inclus dans le champ de données d'une transaction Blockchain.
- Les règles du jeu, correspondantes aux différents coups permis dans un état du plateau et aux conditions de victoire du jeu.
- Le moteur de jeu, qui pourrait être un ensemble de smart contracts qui régissent une partie. La contrainte importante de ce concept est de pouvoir régler les conflits entre joueurs par rapport aux coups joués.

La FIGURE 6.2 présente une ontologie formalisant un jeu d'échecs Blockchain. Les classes orange reprennent des concepts de notre ontologie générique, les cadres gris des types de données, et les cadres bleus des concepts liés au jeu d'échecs.

En plus de cette ontologie, nous avons également proposé un modèle utilisant BPMN. Le modèle repose sur un fonctionnement événementiel, donc nous devons modéliser chaque circonstance qui peut arriver dans l'application. Au

sein de la section 2.1.1, nous avions présenté ce BPMN en exemple, par le biais de la FIGURE 2.1.

Un des principaux événements liés à l'application est la réalisation d'un coup par un des deux joueurs. Lorsqu'un des joueurs joue son coup, il doit le transmettre à l'autre joueur via la Blockchain. Pour ce faire, l'application encapsule le message du joueur, incluant son coup, au cœur d'une transaction Blockchain. Les deux joueurs attendent alors que la transaction soit incluse dans un bloc et qu'assez de confirmations se soient réalisées. Le temps nécessaire pour cette étape dépend des caractéristiques de l'application et de la Blockchain choisie, et peut varier considérablement, entre l'ordre de la seconde et de l'heure. Une fois la transaction incluse dans un bloc et que ce dernier est propagé à travers le réseau, l'état du jeu en cours est modifié en accord avec le coup joué par le moteur de jeu, et les conditions de victoires sont vérifiées.

Contraintes techniques déduites

Dans un but de simplification de ce premier cas d'application, nous nous limitons ici à quelques contraintes techniques. Les deux modélisations précédemment obtenues permettent de déduire un certain nombre de contraintes techniques, qui s'ajoutent aux contraintes fonctionnelles de la DApp.

Tout d'abord, nous souhaitons sécuriser l'ensemble des coups joués d'une partie à l'aide de la Blockchain. Nous savons qu'une partie d'échecs possède en moyenne une quarantaine de coups au total. Ainsi, une contrainte technique est de pouvoir échanger 40 messages entre les deux joueurs d'une partie, sécurisés par la Blockchain. Notre BPMN montre bien que seuls les deux joueurs sont concernés par les coups réels réalisés.

La latence n'est pas un problème majeur ici, en considérant les parties classiques d'échecs, et non les versions avec des temps de réflexion réduits. Cependant, une latence de plusieurs minutes gênerait les joueurs, puisque notre modélisation considère qu'un coup n'est visible auprès de l'autre joueur qu'après la transaction réalisée.

Services Blockchain adaptés à la DApp

Afin d'échanger les coups d'une partie d'échecs, nous pourrions simplement utiliser une Blockchain avec une latence faible. Cependant, nous avons vu dans la sous-section précédente que les messages sécurisés par Blockchain ne concernent que les deux joueurs d'une partie.

Ainsi, l'utilisation de state-channels est particulièrement adaptée à ce cas : il réduit les frais de transactions, est simple à mettre en place, et réduit la latence du système. Une transaction Blockchain, démarrant la partie au premier coup, ouvre un channel entre les deux joueurs. Ceux-ci peuvent alors échanger des

FIG. 6.3 : Nouveau jeu de B2Expand, LTR. Les joueurs évoluent au sein d'une ville spatiale futuriste.

messages tour à tour. Lorsqu'un des joueurs considère qu'une condition de victoire est remplie, le channel est fermé et la partie terminée. Bien sûr, la condition de victoire n'est alors pas vérifiée par la Blockchain. Il faut alors implémenter un système de *challenge*, qui permet de vérifier les messages transmis en cas de litiges. Un exemple de litige est le suivant. Si le joueur 1 agit comme si le joueur 2 n'avait pas effectué d'action avant un certain temps, il pourrait effectuer une transaction prétendant que le joueur 2 n'a pas joué. Le jeu doit donc octroyer au joueur 2 un délai de challenge pour qu'il puisse prouver qu'il a bien joué.

6.2.3 Cas d'étude : *Light Trail Rush* (LTR)

Light Trail Rush (LTR) est un jeu développé par le studio B2Expand. Il s'agit d'un jeu multijoueurs de *brawling*, ou combat entre vaisseaux, dans l'espace, dont une image est montrée FIGURE 6.3. La conception du jeu a, dès le début, pris en compte la technologie Blockchain. En effet, l'entreprise souhaitait que les objets achetables du jeu soient vendus par le biais de cette technologie.

Un premier prototype a été réalisé sous forme de smart contract sur la Blockchain Ethereum (Version Muir Glacier). Les problèmes de latence et les frais liés à toute action d'écriture prohibent l'usage de cette solution, en tous cas avec la version courante de Ethereum. Cela limite donc les fonctionnalités Blockchain que nous pouvons intégrer dans le jeu.

L'usage d'une sidechain ouvre de nouvelles perspectives qui devraient permettre de s'affranchir des problèmes cités ci-dessus. Une des solutions étudiées

est la sidechain Matic [137], qui permet de bénéficier des avantages d'Ethereum tels que la fiabilité tout en s'affranchissant des délais.

Assets Blockchain

Un *asset* de jeu vidéo est un élément du jeu. Il peut s'agir d'un élément graphique cosmétique (des *skins* que les joueurs peuvent posséder), des niveaux, des objets, des sons, des modèles 3D, etc. Les développeurs d'un jeu vidéo ont plusieurs intérêts à lier les assets de leurs jeux à la Blockchain. En effet, l'utilisation d'assets Blockchain permet d'assurer la transparence du contenu des assets aux joueurs, de garder automatiquement un historique complet des interactions avec cet asset, ou encore de permettre aux joueurs de se les acheter, vendre, ou de se les échanger en cryptomonnaie.

Initialement, le jeu LTR souhaite proposer des skins Blockchain aux joueurs. Pour cela, nous avons conçu des smart contracts implémentant l'interface ERC-721. Ce standard permet de décrire les interactions nécessaires avec des assets uniques, des NFT.

Contrats ERC-721

Plusieurs smart contracts ont été développés pour assurer cette fonctionnalité. Nous avons dans un premier temps testé ces contrats en local avec l'aide de Truffle, puis sur un Testnet Ethereum, puis nous les avons déployés sur le MainNet Ethereum. La liste exhaustive des contrats écrits est la suivante :

- NFT_Token, le contrat, implémentant le standard ERC-721, où sont stockés les assets Blockchain du jeu,
- NFT_Factory, un contrat permettant à la fois de *minter* les assets du contrat NFT_Token, ainsi que d'assurer la gestion des différents types d'assets du jeu,
- NFT_B2E_Listing, un contrat offrant des fonctionnalités additionnelles facilitant la lecture des données des deux premiers contrats,
- Helpers, une librairie offrant des fonctions auxiliaires (telles que des fonctions de conversion entre différents types de données),
- ProxyContractForBurn, un contrat permettant de définir les paramètres économiques de nos contrats,
- ProxyForKYCWhitelistEveryoneIsWhitelisted, une implémentation permettant de passer outre les conditions de *Know your Customer* (KYC),
- et ProxyForKYCWhitelistOnlySpecifiedPeopleAreWhitelisted, une implémentation demandant d'être sur liste blanche afin d'assurer le KYC. Dans ce cas, seuls les personnes sur liste blanche peuvent acheter un NFT à

B2Expand. Les autres personnes peuvent toujours utiliser le marché secondaire, puisque seules les ventes directes effectuées par B2Expand sont concernées par les règles de KYC.

Contrat	**Adresse MainNet**
NFT_Token	0x3EB63d51264CE35b7e3320AE2E70813e264C610e
NFT_Factory	0x5ee0900fa78DC40279cC7372884f1A2eE515F876
NFT_B2E_Listing	0x033671B4d777Bd8eAF6d391D48422948504EB6a9
Helpers	0xEACc5882757e9A77087c19f5B9AFb9B5Bb542499
ProxyContractForBurn	0xa50d48bC3c704a00DBa9C6448B5DdF1C962EE74D
ProxyForKYCWhitelist EveryoneIsWhitelisted	0x494f347FD736Ac7CB9411E0c030C58939f6817dA
ProxyForKYCWhitelist OnlySpecifiedPeople AreWhitelisted	0xE3091EdB1DDECd82624c4AB7AC1eE386caAFC59a

Tab. 6.1 : Adresses MainNet Ethereum des différents smart-contrats. Ces contrats sont vérifiés sur le site `https://etherscan.io`.

Le contrat principal gère les assets ERC-721 en eux même. Ainsi, il implémente en particulier l'interface ERC-721. Cela signifie que chacune des fonctions du standard est implémentée dans ce contrat. En particulier, les fonctions les plus importantes sont :

- `function balanceOf(address _owner) external view returns (uint256) ;`
  Cette fonction permet de vérifier le nombre de nos assets qu'une certaine adresse Ethereum possède,
- `function ownerOf(uint256 _tokenId) external view returns (address) ;`
  Cette fonction permet de connaître l'adresse Ethereum qui possède un asset particulier (identifié par son « *tokenId* »),
- `function safeTransferFrom(address _from, address _to, uint256 _tokenId) external payable ;`
  Cette fonction permet de transférer un asset à une adresse Ethereum. Pour appeler cette fonction, le contrat vérifie que l'utilisateur possède bien l'asset, ou qu'il a l'approbation de la personne qui possède l'asset de le gérer. Le système d'approbation de gestion des assets est une autre fonctionnalité des contrats ERC-721 que nous ne détaillons pas dans ce manuscrit.

En plus d'une implémentation de l'interface ERC-721, ce contrat possède des fonctionnalités additionnelles que nous ne détaillerons pas ici, telles que la possibilité de regrouper plusieurs assets afin de transférer l'ensemble de ces assets en une seule transaction.

Un autre contrat important est le contrat *NFT_Factory*, qui assure le suivi des types d'assets que possède chaque joueur, ainsi que la création de nouveaux assets. Ce contrat implémente, entre autres :

- `function createNewOptionID(uint256 nbAssetMaxToMint, address _minter, string[] memory keys, string[] memory values) internal ;`
  Cette fonction permet aux développeurs de LTR, ou techniquement à n'importe qui d'autre, de créer de nouveaux types d'assets compatibles avec le jeu. Cependant, cette fonction est déclarée `internal`, ce qui signifie qu'elle ne peut être appelée que depuis une autre fonction de ce contrat. En effet, l'économie du jeu demande de dépenser des Nexium afin de créer un asset, ce qui est appelé un « brûlage » de Nexium et déclenche l'appel de la fonction suivante, `receiveApproval`,
- `function receiveApproval(address _from, uint256 _value, address _token, bytes memory _extraData) public ;`
  Ainsi, pour créer un asset, un utilisateur appelle le contrat Nexium avec des données pointant vers ce contrat. Cela déclenche l'appel à cette fonction, qui appellera par la suite la fonction précédente `createNewOptionID`,
- `function setTokenURI(uint256 tokenID, string memory _tokenURI) public ;`
  Cette fonction permet d'associer un certain URI à un token, cet URI référençant les métadonnées associées à cet asset tel que des images, un modèle 3D, un nom, etc.
- `function mint(uint256 _optionId, address _toAddress) public ;`
  Enfin, cette fonction permet de produire un nouvel asset à partir d'un type d'asset existant. Cette fonction est par exemple appelée lors de la vente d'un nouvel asset sur notre magasin OpenSea.

Les assets sont ainsi disponibles à l'achat et la vente sur la plateforme OpenSea. Le marché primaire des assets s'effectue sur notre magasin OpenSea personnalisé, dont une capture d'écran est disponible FIGURE 6.4.

Une fois un asset LTR acheté, un joueur peut alors se diriger sur notre interface utilisateur de gestion de ses assets. Une étape essentielle afin de pouvoir utiliser l'asset en jeu est de lier le compte de jeu et l'adresse Ethereum du joueur. L'interface permettant ce lien est montrée en FIGURE 6.5.

6.2.4 Mise en correspondance entre Ethereum et Matic

Les assets Blockchain de ce jeu ont tout d'abord été pensés pour fonctionner sur la Blockchain Ethereum. Les raisons de ce choix sont multiples :

- Disponibilité des ressources d'aide au développement,
- Maturité du projet par rapport à la concurrence,

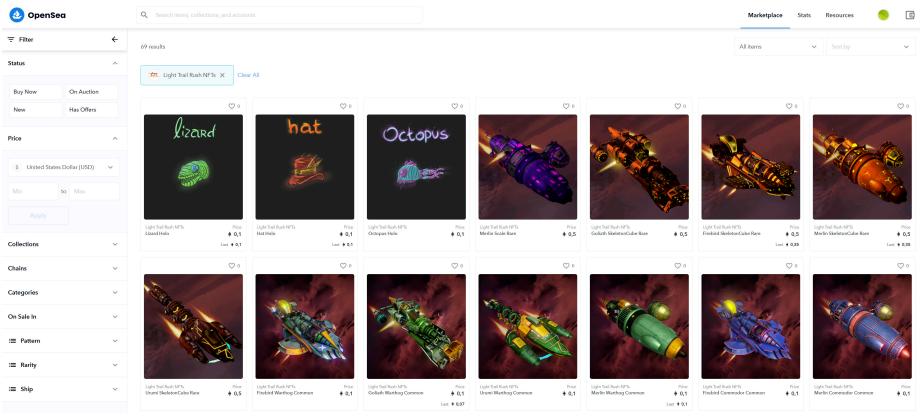

FIG. 6.4 : Store sur le site `https://opensea.io` du jeu LTR.

- Évolutivité de la plateforme.

Cependant, après avoir développé les contrats souhaités, l'entreprise avait pour objectif d'intégrer davantage la Blockchain aux fonctionnalités de jeu. Pour cela, nous avons besoin de réaliser certaines transactions Blockchain en temps réel dans le jeu. Cela représente un coût et une latence des transactions prohibitifs avec Ethereum.

Une étude de différentes solutions de scalabilité des transactions a alors été effectuée. Pour simplifier l'ontologie obtenue, nous avons tout particulièrement étudié Matic, une *sidechain* d'Ethereum. Cependant, la méthodologie que nous avons définit dans le chapitre 3 demande de formaliser l'ensemble des services Blockchains voulus. En effet, en formalisant uniquement Ethereum et Matic, nous ne pouvons pas nous assurer que Matic soit la solution de scalabilité la plus adaptée à notre cas d'application. Formaliser l'ensemble des sidechains disponibles, les state-channels, les multi-chaînes et les *rollups* permettrait de valider le choix effectué.

Comparaison entre Ethereum et Matic

Le tableau 6.2 montre une comparaison d'Ethereum et Matic.

Déploiement de contrats ERC-721 sur Matic

La Blockchain Matic supporte les smart contracts via le même mécanisme qu'Ethereum, puisqu'elle supporte l'EVM [108]. Cela signifie que les contrats écrits pour Ethereum, dans le langage Solidity, peuvent également être déployés sur Matic sans aucune modification. Si l'on souhaite déployer un contrat sur Matic, il nous suffit alors de reprendre les scripts de déploiement utilisés pour

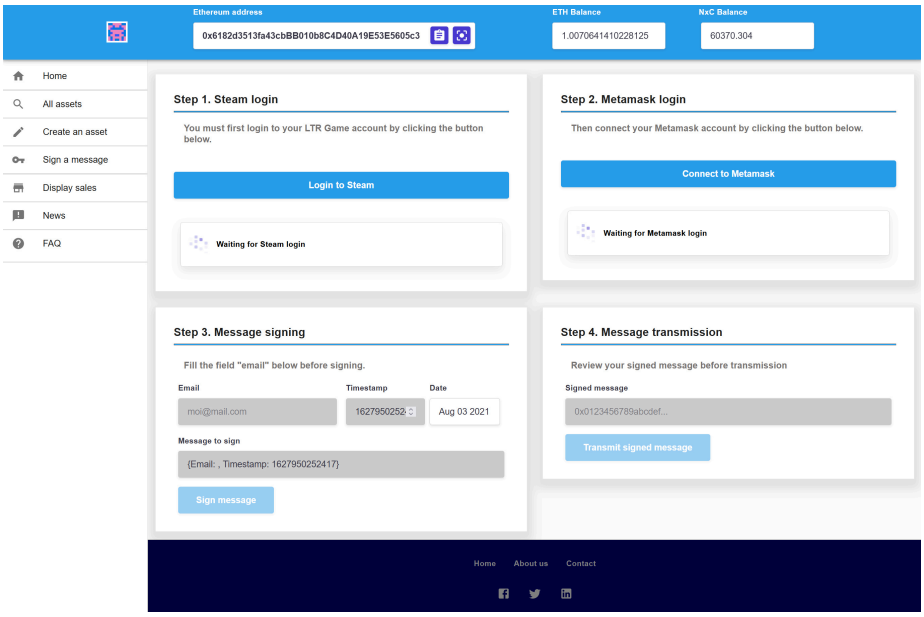

FIG. 6.5 : Page de signature Ethereum de l'adresse email du compte du joueur. Le joueur doit se connecter sur son compte Steam, puis sur son portefeuille Ethereum avec Metamask. Ensuite, il peut signer un message donné et transmettre à B2Expand cette signature pour lier son adresse Ethereum avec son compte de jeu.

Ethereum, et de simplement changer le réseau sur lequel on effectue les transactions dans les configurations de Truffle.

Dans le cas de contrats ERC-721, nous avons le choix :

- Soit nous laissons Matic déployer un contrat ERC-721 standard pour nous. Nous devons alors simplement faire une demande à Matic pour faire correspondre notre propre contrat sur Ethereum avec un nouveau contrat ERC-721.
- Soit nous déployons notre contrat sur Matic, puis nous demandons à Matic de faire correspondre ces deux contrats.

Ici, la deuxième solution est légèrement plus complexe, mais nous permet de garder les fonctionnalités additionnelles de notre contrat, telles que la possibilité de regrouper plusieurs assets afin de transférer l'ensemble de ces assets en une seule transaction. Les contrats déployés par Matic ne contiennent pas ce genre de fonctionnalités non-standards, donc cette première option ne permet pas de les garder.

Caractéristique	Ethereum	Matic
Temps de Bloc	14s	1-2s
Coûts par transaction	1-100€	0.01-1€
Algorithme de consensus	PoW (PoS prévue)	DPoS
Checkpointing	Non	Régulier vers Ethereum
Risques de censures	Très faibles	Modérés
Finalité faible	Quelques minutes	Quelques secondes
Finalité totale	Quelques heures	1 semaine

Tab. 6.2 : Comparaison entre certaines caractéristiques d'Ethereum et de sa sidechain Matic.

Transition entre les deux Blockchains

Grâce à notre ontologie, nous pouvons décrire avec plus de détails les différences techniques du passage d'une des deux Blockchains à l'autre. En effet, dans les sections 5.1.1 et 5.2, nous avons formalisé les deux Blockchains, ainsi que certaines règles contraignant le fonctionnement d'Ethereum et de Matic. Grâce à notre méthodologie, nous avons pu déduire qu'utiliser Matic en tant que solution de scalabilité pour Ethereum était la meilleure solution pour notre application.

Nous avons ainsi modélisé les différents concepts d'Ethereum et de Matic au sein de notre ontologie. Nous avons également écrit les règles SWRL correspondantes aux interactions entre ces deux Blockchains, avec les règles 3 à 6, décrites dans les paragraphes Règle 3 à Règle 6.

Ces règles contraignent les concepts de transactions Matic et Ethereum, et incluent par exemple les concepts de checkpoints et de hashs de transactions.

Nous avons réussi à inférer des correspondances entre des transactions Matic et les checkpoints Matic qui valident ces transactions. Ces correspondances permettent de mieux comprendre les différences sémantiques entre Matic et Ethereum. En revanche, nous n'avons pas obtenus d'autres inférences permettant de mieux définir ces différences. Cela peut signifier que nous n'avons pas assez contraint les concepts modélisés, ou bien que les relations entre ces concepts sont trop larges pour être quantifiées sans faire appel à un nombre trop important de concepts intermédiaire. En effet, l'objectif n'est pas de modéliser manuellement les différences entre les deux Blockchains, mais bien de les obtenir à partir de raisonnement sur des règles simples.

Néanmoins, cette ontologie nous permet tout de même de déduire d'autres contraintes. Un exemple simple d'une telle contrainte est que nos utilisateurs ont besoin d'avoir un compte Blockchain Matic. Heureusement, l'obtention de ce compte est simple puisqu'il suit les mêmes standards que ceux d'Ethereum.

Ainsi, tout utilisateur d'Ethereum possède automatiquement un compte Matic associé.

Finalement, pour utiliser Matic au sein de LTR, notre méthodologie propose de procéder comme suit :

1. Au sein du jeu développé avec Unity, remplacer tous les appels à la Blockchain Ethereum par des appels à la Blockchain Matic. Cela fonctionne puisque Matic repose sur le même écosystème qu'Ethereum.
2. De même, sur la DApp de gestion de NFT développée avec les technologies du web, il est possible de demander aux utilisateurs de se connecter sur Matic avec les mêmes outils qu'ils utilisent habituellement pour Ethereum.
3. Le *mint* des assets, réalisés par B2Expand, se réalise uniquement sur Ethereum, donc cette partie-là ne change pas de fonctionnement.
4. Cependant, le transfert d'assets entre joueurs se ferra dorénavant sur Matic, réduisant les coûts associés.

La FIGURE 6.6 présente un modèle BPMN décrivant un exemple de cycle de vie d'un asset LTR utilisant la solution Matic en tant que sidechain d'Ethereum.

Ici, B2Expand créé et mint l'asset uniquement sur Ethereum, puis le transfert vers le contrat maître de Matic par une transaction Ethereum. L'utilisateur 1 achète alors l'asset sur Matic, et peut le transférer avec des coûts très faibles à l'utilisateur 2. L'utilisateur 2 souhaite récupérer l'asset sur Ethereum. En effet, il est possible qu'il veuille utiliser cet asset sur une plateforme qui ne supporte pas l'utilisation de Matic. Il demande alors le retrait de l'asset sur Ethereum et attend la période de challenge.

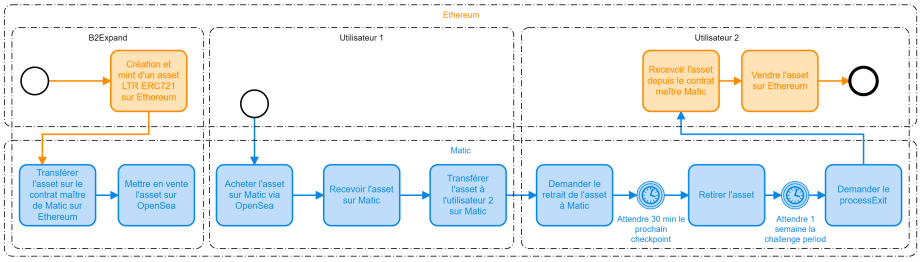

FIG. 6.6 : BPMN montrant le transfert d'assets LTR entre Ethereum et Matic.

6.3 Validation de notre prototype

Une fois les différents scénarios d'applications implémentés, nous devons évaluer les prototypes obtenus. Ces évaluations nous permettent de nous assurer que les étapes de notre méthodologie ont bien été respectées et que leurs hypothèses sont vérifiées. Cela nous permettra également de juger les contributions de notre méthodologie pour la recherche sur l'interopérabilité des systèmes Blockchain, et ainsi que nous répondons bien aux questions de recherches posées lors de l'établissement de notre problématique.

Cependant, afin d'être pertinentes, nos évaluations doivent être établies selon des critères bien choisis et exhaustifs. Dans le chapitre suivant, nous détaillons ainsi dans un premier temps les critères retenus, puis les résultats de nos évaluations.

6.4 Synthèse

Dans ce chapitre nous étudions deux cas concrets d'application de nos recherches dans le monde industriel. Nous appliquons dans un premier temps nos recherches au cas d'un jeu d'échecs Blockchains. En suivant la méthodologie présentée dans le chapitre 3, nous construisons une ontologie formalisant les concepts de ce jeu. Nous modélisons les processus du jeu avec BPMN, déduisons des contraintes techniques et fonctionnelles de l'application, puis choisissons des services Blockchain adaptés à celle-ci.

Dans un second temps, nous appliquons nos recherches au jeu vidéo Blockchain de B2Expand, LTR. Ce jeu Blockchain, prévu initialement pour utiliser la Blockchain Ethereum, peut également être utilisé avec d'autres plateformes, telles que Matic. À partir des résultats obtenus dans la section 5.1.4, nous choisissons d'utiliser Matic en tant que sidechain d'Ethereum, ce qui permet de réduire les coûts utilisateur de la DApp. En effet, les choix de développement impactent les caractéristiques techniques de l'application, les différents processus de fonctionnement, ou encore les coûts pour les développeurs ou les utilisateurs.

De plus, plusieurs idées pourraient être étudiées pour le futur de LTR, que nous détaillons ci-dessous.

Utilisation de la Blockchain au sein du jeu

Tout d'abord, grâce à nos recherches, d'autres mécaniques de jeu utilisant la technologie Blockchain peuvent être pensées. Le monde dans lequel évoluent les joueurs pourrait en effet dépendre des comptes Blockchain des joueurs.

Réciproquement, les actions de chaque joueur pourraient mener à des écritures dans la Blockchain. Un exemple classique est l'obtention de monnaie ou d'assets en cas de succès de jeu.

Approche multi-Blockchain

Aujourd'hui, nous avons étudié les possibilités de transition d'une Blockchain à une autre pour LTR, et les conséquences de cette transition. Cependant, l'approche modulaire de la conception du jeu n'empêche pas l'implémentation de l'intégration de plusieurs Blockchains différentes.

Cela permettrait, par exemple, de laisser le choix aux joueurs des technologies utilisées. Dans ce cas, nous approchons également les besoins d'interopérabilité entre les Blockchains. Une problématique est de s'assurer qu'un même asset peut être représenté de manière totalement équivalente entre deux Blockchains.

Chapitre 7

Évaluations de nos contributions

Afin de s'assurer de l'intérêt de recherche de nos contributions, nous devons les évaluer. Pour cela, nous décrivons nos critères d'évaluation puis nos résultats.

7.1 Évaluation de nos deux contributions de recherche

Nous avons détaillé dans les chapitres 4 et 5 les constructions et applications de recherche de notre méthodologie et de notre ontologie. Il est alors important de les évaluer afin de s'assurer que nous répondons bien à la problématique initiale d'amélioration de l'Interopérabilité des Systèmes Blockchain. Pour cela, nous définissons un certain nombre de critères d'évaluation et jugeons les contributions via ces critères.

7.1.1 Critères d'évaluation

Évolutivité

Le domaine de la Blockchain évoluant rapidement, il est primordial de s'assurer que les contributions de cette thèse ne deviennent pas obsolètes avec l'arrivée d'une nouvelle technologie. L'évolutivité de nos contributions consiste en l'élaboration d'un protocole prévoyant les mises à jours de celles-ci en fonction des changements que subit l'écosystème Blockchain.

Pour notre méthodologie, la mesure de l'évolutivité consistera à étudier les changements potentiels des différentes étapes qui la composent.

Pour notre ontologie, l'évolutivité consiste à s'assurer que la formalisation de nouveaux concepts reste possible, sans considérablement impacter les concepts déjà modélisés.

Facilité d'application de la méthodologie

Si notre méthodologie doit être utilisée par de nombreux chercheurs, il faut s'assurer qu'elle soit facile à prendre en main. Cela signifie qu'une personne non experte peut comprendre la méthodologie et ses objectifs. De plus, les différentes étapes ne doivent pas demander des compétences expertes relativement aux outils utilisés.

Facilité d'utilisation de l'ontologie

De même, il devrait être facile d'étendre une ontologie si l'on souhaite qu'elle soit utilisée. Cette facilité peut par exemple provenir d'aides à la formalisation de certains concepts usuels des DApps.

Avis de la communauté Blockchain

Un autre critère pertinent est la réception de nos contributions au sein de la communauté Blockchain. Si nos contributions sont pertinentes, alors d'autres concepteurs de DApp vont réutiliser notre méthodologie et notre ontologie. Il est également possible de recevoir des retours par le biais d'une enquête publique afin d'améliorer nos contributions.

Synthèse

Le tableau 7.1 synthétise les différents critères d'évaluation choisis.

Critère	**Opérationnalisation**
Évolutivité	Résilience aux apparitions de nouveaux services Blockchain
Facilité application méthodologie	Niveau de connaissances à avoir pour l'appliquer
Facilité utilisation ontologie	Niveau de connaissances à avoir pour formaliser une nouvelle DApp
Évaluations des pairs	N.A.

Tab. 7.1 : Critères d'évaluation de nos contributions de recherche.

7.1.2 Résultats

Évolutivité

Pour notre méthodologie, les différentes étapes permettent bien de garantir une certaine évolutivité. Si un nouveau service Blockchain est développé, alors il

faut simplement le prendre en compte dans la comparaison avec les contraintes techniques de l'application. Cependant, il faudra possiblement ajouter une étape à notre méthodologie permettant l'utilisation d'autres outils de formalisation. En particulier, une formalisation des enjeux financiers associés à une DApp pourrait devenir pertinente dans le cadre d'applications avec de nombreux acteurs, comme pour les grosses entreprises. Ce besoin n'est pas encore présent, mais s'il le devient, des outils de la théorie des jeux pourraient prendre place au sein de notre méthodologie. En plus de l'ajout de l'étape, l'unique adaptation à notre méthodologie sera alors la prise en compte des résultats fournis par cet outil pour en déduire des contraintes techniques complémentaires.

Notre ontologie est également évolutive pour plusieurs raisons. Tout d'abord, si un nouveau type de service Blockchain apparaît, nous pouvons y associer des propriétés de latence et coûts comme les autres services. La modélisation est modulaire : nous pouvons définir quels composants forment une DApp donnée. D'autres règles SWRL peuvent également être écrites si nous avons besoin de formaliser des contraintes plus complexes par rapport à un service Blockchain particulier.

De plus, d'autres chercheurs peuvent ajouter des concepts, ou étendre notre ontologie avec de nouvelles ontologies Blockchain. De même que pour notre méthodologie, l'ajout de concepts relatifs aux enjeux économiques de différents acteurs pourrait être considéré.

Facilité d'application de la méthodologie

La facilité d'application de la méthodologie à une nouvelle DApp est relative. Tout d'abord, il est nécessaire de bien comprendre les outils utilisés par la méthodologie et leurs objectifs. Un certain nombre de connaissances expertes sont donc nécessaires pour formaliser une DApp entièrement. Cependant, une fois une DApp modélisée, il est possible de reprendre la modélisation pour l'adapter à une nouvelle. Des chercheurs ne maîtrisant pas ces outils peuvent donc se reposer sur les travaux effectués sur d'autres DApp.

Par exemple, notre ontologie Blockchain inclus des Blockchains Patterns permettant de facilement modéliser de nouvelles DApps qui utilisent des Patterns de développement connus dans le domaine.

Cependant, la dernière étape de notre méthodologie, relative à la mise en correspondance des contraintes techniques et les services Blockchains adaptés pour répondre à ces contraintes, se fait manuellement. Une de nos perspectives est de simplifier l'application de cette étape en proposant un système d'évaluation multicritères.

Facilité d'utilisation de l'ontologie

Comme mentionné précédemment, nous avons intégré des formalisations partielles de Blockchain Patterns communs. Cela permet de facilement utiliser l'ontologie pour formaliser de nouveaux cas d'usages de celle-ci.

Pour formaliser des contraintes plus complexes, des connaissances en SWRL sont nécessaires. Il est possible de s'aider des règles existantes au sein de l'ontologie pour en écrire de nouvelles, mais l'expressivité de ce langage peut rendre difficile cette tâche pour des problèmes complexes.

Évaluations des pairs

Nous prévoyons soumettre à publication notre ontologie prochainement afin de pouvoir recueillir les retours des chercheurs du domaine, mais également des développeurs de DApp par le biais d'une enquête publique.

Synthèse

Le tableau 7.2 synthétise les différents résultats obtenus.

Critère	**Résultats**
Évolutivité	La méthodologie et l'ontologie sont évolutives
Facilité application méthodologie	Méthodologie simple d'application, mais nécessite des connaissances métiers pour certaines tâches
Facilité utilisation ontologie	L'ontologie guide les utilisateurs grâce aux Blockchain Patterns
Évaluations des pairs	Les résultats seront obtenus après publication

Tab. 7.2 : Résultats obtenus sur l'évaluation de nos contributions de recherche.

7.2 Évaluation de notre application industrielle

Nous avons détaillé dans le chapitre 6 l'application de nos travaux de recherche au domaine du jeu vidéo, avec le jeu vidéo LTR de B2Expand. Il faut évaluer cette application afin de s'assurer d'une part que nous avons correctement appliqué nos contributions, et d'autre part que nos contributions nous ont bien permis d'obtenir les résultats souhaités. Pour cela, nous allons définir un certain nombre de critères d'évaluations et jugeons l'application industrielle obtenue via ces critères.

7.2.1 Critères d'évaluation

Évolutivité

Comme nous avons pu le voir dans ce mémoire, la technologie Blockchain, et les écosystèmes associés, sont très innovants. Cela amène à une difficulté commune lors du développement de DApp : de nouvelles solutions techniques apparaissent régulièrement, pouvant rendre obsolètes les développements réalisés.

Un des critères d'évaluation de notre application industrielle est donc l'évolutivité. Nous devons nous assurer que la solution adoptée ne pourra pas être rendue obsolète rapidement sans que l'on ait les moyens de nous adapter à celle-ci.

Modularité

La modularité d'une application réfère à l'architecture de cette dernière. La maintenance d'une application modulaire est davantage aisée que pour une application non modulaire. Nous avons vu en section 2.3.2 qu'une architecture par microservices était particulièrement adaptée pour les DApps.

Concrètement, notre application aura une bonne modularité s'il est possible d'adapter les différents composants de l'application facilement. Ce concept est donc complémentaire avec l'évolutivité.

Utilisation de l'ontologie

L'application industrielle servant à valider les contributions de thèse, il est important que nous utilisions l'ontologie développée au maximum. Cela permettra en effet d'étudier les possibilités que notre ontologie offre en termes de formalisation de DApp, mais également ses limitations.

En particulier, nous devons évaluer les fonctionnalités de notre ontologie au regard de l'ensemble des fonctionnalités de cet outil, en termes d'expressivité et de cohérence.

Utilisation des modèles BPMN

De façon analogue avec le critère précédent, nous devons nous assurer que notre DApp tire bien parti des modèles BPMN construits. Cela signifie que nous avons bien réussi à modéliser l'ensemble des interactions de l'application.

Afin de valider notre choix d'utilisation de BPMN comme outil de modélisation de notre DApp, nous devons analyser ce que pourrait apporter l'utilisation d'outils de modélisation additionnels.

Satisfaction des contraintes fonctionnelles

L'objectif principal de notre méthodologie est de traduire les spécifications d'une DApp en liste d'instances de services à utiliser pour construire l'application. Ainsi, nous devons évaluer le degré de satisfaction des contraintes fonctionnelles présentent dans les spécifications initiales.

Satisfaction des contraintes technologiques

Un autre objectif de la méthodologie est la satisfaction des contraintes technologiques obtenues à partir des modélisations par BPMN et par l'ontologie construite.

Concrètement, si ce critère d'évaluation n'est pas pleinement satisfait, cela signifie que l'application finale ne fonctionnera pas de façon optimale, puisque les contraintes de scalabilité, de latence ou d'intégration des services ne seront pas bien respectées.

Évaluation par les utilisateurs

Un dernier critère intéressant pour notre évaluation est de prendre en compte les avis d'experts dans le développement de jeux Blockchain et des utilisateurs de notre jeu. Cela nous permettra de comparer les résultats obtenus avec ce qu'ils pensent est la manière optimale de développer une telle DApp, et de s'assurer que l'expérience utilisateur est optimale.

Synthèse

Le tableau 7.3 synthétise les différents critères d'évaluation choisis.

Critère	**Opérationnalisation**
Évolutivité	Résilience aux apparitions de nouveaux services Blockchain
Modularité	Possibilité de mise à jour de fonctionnalités sans en modifier d'autres
Utilisation ontologie	Degré de formalisation par ontologie
Utilisation BPMN	Degré de modélisation par BPMN
Contraintes fonctionnelles	Nombre de contraintes prises en compte
Contraintes technologiques	Nombre de contraintes prises en compte
Évaluation utilisateurs	N.A.

Tab. 7.3 : Critères d'évaluation de notre application industrielle.

7.2.2 Résultats

Évolutivité

Notre application industrielle dépend de la Blockchain sur deux points principaux. Tout d'abord, le jeu LTR, développé avec l'aide du moteur de jeu Unity, doit avoir accès en lecture aux informations contenues au sein des smart contracts déployés. En effet, nous devons savoir, pour chaque joueur, à quels assets graphiques il a accès. La connexion à la Blockchain se réalise par le *package* Nethereum pour Unity, spécifique à l'écosystème Ethereum. Ainsi, nous dépendons de cet écosystème pour notre jeu. Cependant, une partie importante des projets étant développés aujourd'hui restent compatibles avec cet écosystème, afin de faciliter la prise en main et la transition vers leur solution.

Un second point concerne l'application de gestion des NFT d'un joueur. Cette application à besoin d'un accès à la Blockchain en écriture. En effet, chaque joueur peut se connecter sur son compte Metamask pour réaliser des transactions sur notre DApp. Ici, nous dépendons également de l'écosystème Ethereum, mais de manière moins importante. En effet, l'intégration avec les technologies du web est une priorité des nouveaux projets. Ainsi, il sera facile de réaliser la transition vers une autre solution si besoin. Il nous suffira principalement d'appeler des API différentes.

Pour ce qui est du processus de transition entre Ethereum et Matic, notre application se doit d'être évolutive afin de s'adapter aux changements des différents projets tiers impliqués. En effet, les protocoles d'Ethereum et de Matic sont toujours en développement à l'heure actuelle. Ethereum vise ainsi le passage à la PoS courant 2022, et Matic développe un écosystème propice aux multi-chaînes. Pour cela, en plus des éléments détaillés dans le point précédent, l'application étant modulaire, nous pouvons développer des modules de tests appelant les API des protocoles en développement. Une fois les spécifications figées, nous pouvons alors valider le bon fonctionnement des nouveaux protocoles. Ce point est davantage détaillé dans le paragraphe Modularité des plateformes Blockchain utilisées.

Modularité

Modularité des smart contracts Les smart contracts de gestion des assets de LTR sont en partis modulaires. En effet, le *Design Pattern* de proxy permet de changer certaines fonctionnalités de l'application en déployant de nouveau seulement le smart contract gérant cette fonctionnalité. Les autres contrats ne sont alors pas impactés par ce changement. Un exemple d'application possible de cette caractéristique est le changement de la formule du coût de création d'assets. La création d'assets LTR nécessite le brûlage de la cryptomonnaie de B2Expand, le Nexium. Nous pouvons adapter le nombre de Nexium à brûler en fonction de la conjonction crypto-économique à un certain instant, c'est-à-dire les cours des marchés de cryptomonnaies.

Modularité des interfaces utilisateur L'ensemble des interfaces utilisateur supportant la Blockchain permet d'utiliser l'ensemble des fonctionnalités requises : appel d'un smart contract spécifique, lecture des données sur la Blockchain, connexion au compte d'une Blockchain, gestion de différents réseaux différents, etc.

Ainsi, si nous développons un nouveau jeu qui n'utilise pas Unity comme le fait LTR, le nouveau moteur de jeu pourra réutiliser l'ensemble des autres fonctionnalités de nos applications. Cela fonctionne bien sûr à condition qu'il soit possible de se connecter à la Blockchain avec ce nouveau moteur. C'est pourquoi les différentes interfaces utilisateur sont modulaires.

Modularité des plateformes Blockchain utilisées Comme indiqué précédemment, les outils de connexion à la Blockchain supportent différents protocoles. Ainsi, les outils de connexion à Ethereum ou à Matic sont les mêmes. La modularité des plateformes Blockchain utilisées revient ainsi à la modularité des fichiers de configuration utilisés, ainsi que les différentes contraintes techniques.

Par exemple, si nous devions passer de Matic vers une autre plateforme, il faudra s'assurer grâce à notre méthodologie que cette nouvelle plateforme est bien adaptée aux contraintes de l'application. Si c'est le cas, la transition vers une nouvelle Layer Two se fait de façon similaire au passage d'Ethereum vers Matic.

Utilisation de l'ontologie

L'ontologie de nos contributions de recherche nous a permis, entre autres, de formaliser la transition entre Ethereum et Matic pour notre DApp. Détailler les différents concepts liés à ces deux plateformes permet alors de s'assurer de la pertinence des programmes écrits.

Nous avons également utilisé les règles SWRL relativement aux contraintes de coûts et de latence pour analyser les trois cas d'usage possibles de notre application. L'utilisation de SWRL permet une plus grande expressivité de ces contraintes.

Utilisation des modèles BPMN

Nous avons pu modéliser certaines interactions de l'application par des modèles BPMN. Principalement, nous avons analysé le cycle de vie d'un asset LTR déployé sur Ethereum, et utilisé sur sa sidechain Matic.

Dans le cas de notre DApp, les interactions actuelles entre les joueurs de LTR restent très simples, donc les modélisations par BPMN ne sont pas toujours les plus intéressantes par rapport à l'usage que pourraient avoir d'autres DApp.

De même, pour notre cas d'usage, l'utilisation d'autres outils de modélisation tels que présentés au sein de la section 2.1 ne permettraient pas de mieux décrire davantage l'application. La formalisation des smart contracts par des machines à états finis est une exception, mais cette approche reste complexe à mettre en place de façon générique.

Satisfaction des contraintes fonctionnelles

La contrainte fonctionnelle de base de l'application développée est la gestion de NFT pour le jeu LTR. Cette contrainte est pleinement validée, puisque l'ensemble des standards permettant cette gestion sont implémentés. De plus, nous avons ajouté certaines fonctionnalités spécifiques à notre DApp, telles que le brûlage de Nexium ou le groupage d'assets.

La transition à l'utilisation de Matic permet aussi de préparer la satisfaction de nouvelles contraintes fonctionnelles. Par exemple, il sera possible de réaliser des écritures sur la Blockchain lors d'événements liés au jeu, tels qu'une victoire d'un joueur ou de la collecte d'éléments dans le jeu qui pourraient donner au joueur un NFT correspondant. Cependant, ces nouvelles fonctionnalités ne sont pas encore implémentées au sein du jeu, donc il ne nous est pas possible de les valider complètement.

Satisfaction des contraintes technologiques

De même que pour la satisfaction des contraintes fonctionnelles, nous ne pouvons valider la satisfaction de l'ensemble des contraintes technologiques au sein du jeu. Cependant, nous avons réalisé des tests d'intégration pour valider le fait que l'utilisation de Matic ne pose pas de problèmes particuliers à la gestion de nos assets.

Évaluation des utilisateurs

Une fois une version du jeu complétée avec les nouvelles fonctionnalités apportées par nos travaux sera sortie, nous pourrons récolter les retours utilisateur et de nos partenaires industrielles sur notre DApp.

Synthèse

Le tableau 7.4 synthétise les différents résultats obtenus.

Critère	**Résultats**
Évolutivité	La DApp est évolutive
Modularité	La DApp est modulaire sur trois axes : les smart contracts, les interfaces utilisateurs, et les plateformes Blockchain utilisées
Utilisation ontologie	Les formalisations des trois cas nous a permis de choisir quel cas implémenter
Utilisation BPMN	L'outil BPMN permet la compréhension du cycle de vie des assets LTR
Contraintes fonctionnelles	Les contraintes fonctionnelles de gestion de NFT sont validées. Les contraintes fonctionnelles liées à l'utilisation de Matic seront validées une fois le jeu terminé.
Contraintes technologiques	Les contraintes technologiques liées à la gestion de NFT sur Matic sont validées. L'écriture sur la Blockchain depuis le jeu sera validée une fois le jeu terminé.
Évaluation utilisateur	Le prototype sera ouvert pour que tous les utilisateurs puissent tester l'application

TAB. 7.4 : Résultats obtenus sur l'évaluation de notre application industrielle.

Conclusion et perspectives

Conclusions

Les travaux de cette thèse ont abouti à deux contributions majeures améliorant l'interopérabilité des Applications Blockchain Décentralisées, ou DApps. Dans un premier temps, nous avons présenté une méthodologie que les chercheurs et développeurs d'applications Blockchain décentralisées peuvent utiliser afin de faciliter la conception de leurs applications. En effet, cette méthodologie présente différentes étapes permettant d'isoler les caractéristiques techniques associées à une application, et d'associer ces caractéristiques avec des services Blockchain adaptés à l'application. Sans cette méthodologie, il peut être complexe pour un développeur de s'assurer que les services Blockchain qu'il souhaite utiliser au sein de son application répondent bien aux besoins de celle-ci.

Cette première contribution nous a amenés à l'élaboration d'une ontologie Blockchain qui formalise et définit les concepts d'une application Blockchain décentralisée. Cette ontologie étend une ontologie Blockchain existante. Elle définit à la fois des concepts généraux sur le domaine de la Blockchain, ainsi que des propriétés permettant de bien définir une application Blockchain décentralisée. Cette ontologie, associée à des règles SWRL que nous avons définit, permet de déduire automatiquement certaines caractéristiques d'une application en fonction des services Blockchain qu'elle utilise.

En effet, notre méthodologie et notre ontologie permettent bien d'améliorer l'interopérabilité des composants d'une application Blockchain. Les outils de notre méthodologie permettent de formaliser les différents composants et interactions d'une Application Blockchain Décentralisée donnée. L'ontologie proposée aide à cette formalisation en proposant un cadre sémantique à utiliser pour cette formalisation.

L'application industrielle et les résultats déduits permettent alors de valider les propriétés et les contraintes de ces outils. Ils sont en particulier évolutifs et simples d'utilisation, même si certaines connaissances des services Blockchains existants sont nécessaires pour s'assurer d'obtenir une formalisation bien formée.

Ces deux contributions permettent alors de répondre à la problématique initiale de ma thèse : « Quels approches et outils permettraient une amélioration de l'interopérabilité au cœur des Applications Blockchain Décentralisées, et quelles sont leurs contraintes ? »

Nous avons ainsi répondu à l'ensemble des questions de recherche posées. Nos deux premiers chapitres proposent une vue globale des motivations, limites, et état de l'art de l'interopérabilité au sein des Systèmes Blockchains, ce qui répond à notre première question : « Que signifie l'interopérabilité appliquée aux systèmes Blockchains, quelles en sont les motivations, et en quoi est-elle limitée actuellement ? ». Nos contributions de thèse, que nous avons détaillées dans les chapitres 3 et 4 ainsi que récapitulées dans cette conclusion, répondent à la question : « Comment améliorer l'interopérabilité entre les composants d'une application Blockchain décentralisée ? ». Enfin, la dernière question de recherche, « Comment mettre en application nos travaux sur l'interopérabilité des systèmes Blockchain dans un cadre industriel ? », est répondue au sein du cinquième chapitre, lors de l'application de nos recherches à l'industrie du jeu vidéo.

Perspectives de recherche

Cette thèse présente de nombreuses perspectives de recherche. Tout d'abord, à court terme, nous souhaitons diffuser en libre accès notre ontologie. Cela permettra à d'autres chercheurs d'évaluer l'ontologie obtenue, afin de s'assurer par exemple que leur cas d'application est bien compatible avec notre ontologie. En outre, cela nous permettra d'obtenir des retours constructifs sur l'utilisation dans le temps de l'ontologie. Le domaine de la Blockchain évoluant rapidement, nous pourrons nous assurer que les définitions proposées sont bien évolutives. Enfin, cela permettra aux développeurs de services Blockchain d'ajouter une formalisation de leurs propres services, ce qui rendra l'ontologie plus exhaustive.

Nous avons également des perspectives à moyen terme. Tout d'abord, nous nous sommes focalisées comme deuxième contribution à l'élaboration d'une ontologie Blockchain générique. Cependant, notre méthodologie proposée en première contribution à besoin d'autres outils. Par exemple, nous pourrions utiliser les résultats que nous fournit l'ontologie au sein d'un système de recommandation multicritères. Ce système recommandera automatiquement les services Blockchain les plus adaptés à une application, sans avoir besoin d'un avis expert une fois l'application entièrement modélisée. Pour que ce système fonctionne, il faudra alors que le développeur d'une DApp puisse quantifier les compromis qu'il souhaite faire en termes de sécurité, coûts, latence, etc. Une fois ces choix effectués, ils impacteront les poids du système multicritères.

Un autre point que nous pourrions approfondir est la conception d'autres contraintes, telles de nouvelles règles SWRL, ayant pour objectif de différencier les caractéristiques de plusieurs Blockchains. Aujourd'hui, les règles écrites

ne permettent par exemple pas de comparer directement les Blockchains Ethereum et Harmony. Il nous faut en effet modéliser un cas d'application qui utilise chacune de ces deux technologies pour comparer les caractéristiques de ces applications. Pour cela, il faudrait que nous formalisions davantage d'éléments relatifs au fonctionnement théorique d'une Blockchain, tel que le modèle économique. Par exemple, certaines Blockchains fonctionnant par preuve d'enjeu introduisent un *slashing* des fonds : un acteur malveillant peut se faire retirer de la monnaie directement par le protocole. Établir une théorie économique des Blockchains permettra ainsi de formaliser les intérêts de chaque acteur et mieux comparer différentes plateformes.

Enfin, à plus long terme, une dernière perspective de nos recherches sur l'interopérabilité des systèmes Blockchains est l'application de ces recherches à d'autres niveaux d'interopérabilité. Par exemple, le sujet de l'interopérabilité entre les Blockchains est techniquement très intéressant. Notre ontologie pourrait ainsi être étendue pour prendre en compte et formaliser les différentes solutions d'interopérabilité entre Blockchains. Il sera alors possible de modéliser des DApps qui interagissent avec plusieurs Blockchains selon leurs caractéristiques.

Bibliographie

[1] M. Nofer, P. Gomber, O. Hinz, and D. Schiereck, "Blockchain," *Business & Information Systems Engineering*, vol. 59, no. 3, pp. 183–187, Jun. 2017. [Online]. Available : http://link.springer.com/10.1007/s12599-017-0467-3

[2] V. B. al., "A next-generation smart contract and decentralized application platform," 2014. [Online]. Available : https://github.com/ethereum/wiki/wiki/White-Paper

[3] V. Buterin, "Vitalik Buterin on about.me." [Online]. Available : https://about.me/vitalik_buterin

[4] "Huntercoin | XAYA." [Online]. Available : https://xaya.io/huntercoin-legacy/

[5] XAYA, "The ultimate blockchain gaming platform." [Online]. Available : https://xaya.io/downloads/XAYA_White_Paper.pdf

[6] B2Expand, "Light Trail Rush." [Online]. Available : https://lighttrailrush.com/

[7] Dapper Labs, "Cryptokitties. "cryptokitties | collect and breed digital cats!"." [Online]. Available : https://www.cryptokitties.co/

[8] E. Ordano, A. Meilich, Y. Jardi, and M. Araoz, "Decentraland, a blockchain-based virtual world," 2017. [Online]. Available : https://decentraland.org/whitepaper.pdf

[9] C. D. Clack, V. A. Bakshi, and L. Braine, "Smart contract templates : foundations, design landscape and research directions," 2017.

[10] S. Nakamoto, "Bitcoin : A peer-to-peer electronic cash system," Self-Published, Tech. Rep., 2008.

[11] S. A. Abeyratne and R. P. Monfared, "Blockchain ready manufacturing supply chain using distributed ledger," *International Journal of Research in Engineering and Technology*, vol. 5, pp. 1–10, September 2016.

[12] Y. Guo and C. Liang, "Blockchain application and outlook in the banking industry," *Financial Innovation*, vol. 2, p. 1, December 2016.

[13] A. Azaria, A. Ekblaw, T. Vieira, and A. Lippman, "Medrec : Using blockchain for medical data access and permission management," in *International Conference on Open and Big Data (OBD)*, 2016, pp. 25–30.

[14] R. Materese, "Blockchain," Sep. 2019, last Modified : 2021-03-02T08 :49-05 :00. [Online]. Available : https://www.nist.gov/blockchain

[15] "Blockchain Gaming : The Good, the Bad, the Ugly, And All Their Intersections | Hacker Noon." [Online]. Available : https://hackernoon.com/blockchain-gaming-the-good-the-bad-and-the-ugly-h1d92g6k

[16] Journal du Net, "Framework ou infrastructure logicielle : définition et traduction." [Online]. Available : http://web.archive.org/web/20180527230757/https://www.journaldunet.fr/web-tech/dictionnaire-du-webmastering/1203355-framework/

[17] B. Hayes, "Cloud computing," 2008.

[18] C. M. MacKenzie, K. Laskey, F. McCabe, P. F. Brown, R. Metz, and B. A. Hamilton, "Reference model for service oriented architecture 1.0," *OASIS standard*, vol. 12, no. S 18, 2006.

[19] L. Lamport, R. Shostak, and M. Pease, "The byzantine generals problem," in *Concurrency : the Works of Leslie Lamport*, 2019, pp. 203–226.

[20] C. Zhang, P. Dhungel, D. Wu, and K. W. Ross, "Unraveling the bittorrent ecosystem," *IEEE transactions on parallel and distributed systems*, vol. 22, no. 7, pp. 1164–1177, 2010.

[21] Bridgefy, "Offline Messages App & SDK." [Online]. Available : https://bridgefy.me

[22] Z. Zheng, S. Xie, H.-N. Dai, X. Chen, and H. Wang, "Blockchain challenges and opportunities : a survey," *Int. J. Web and Grid Services,*, vol. 14, no. 4, pp. 352–375, 2018.

[23] A. M. Antonopoulos and G. Wood, *Mastering ethereum : building smart contracts and dapps*. O'reilly Media, 2018.

[24] S. Haber and W. S. Stornetta, "How to time-stamp a digital document," in *Conference on the Theory and Application of Cryptography*. Springer, 1990, pp. 437–455.

[25] A. Gervais, G. O. Karame, K. Wüst, V. Glykantzis, H. Ritzdorf, and S. Capkun, "On the Security and Performance of Proof of Work Blockchains," in *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, ser. CCS '16. New York, NY, USA : ACM, 2016, pp. 3–16. [Online]. Available : http://doi.acm.org/10.1145/2976749.2978341

[26] S. King and S. Nadal, "PPCoin : Peer-to-Peer Crypto-Currency with Proof-of-Stake," *Self-published paper*, p. 6, 2012.

[27] L. LAMPORT, R. SHOSTAK, and M. PEASE, "The byzantine generals problem," *ACM Transactions on Programming Languages and Systems*, vol. 4, no. 3, pp. 382–401, 1982.

[28] R. Zhang and W. K. V. Chan, "Evaluation of energy consumption in block-chains with proof of work and proof of stake," in *Journal of Physics : Conference Series*, vol. 1584, no. 1. IOP Publishing, 2020, p. 012023.

[29] D. Larimer, "DPOS Consensus Algorithm - The Missing White Paper — Steemit." [Online]. Available : https://steemit.com/dpos/@dantheman/dpos-consensus-algorithm-this-missing-white-paper

[30] S. M. S. Saad and R. Z. R. M. Radzi, "Comparative review of the blockchain consensus algorithm between proof of stake (pos) and delegated proof of stake (dpos)," *International Journal of Innovative Computing*, vol. 10, no. 2, 2020.

[31] P. Network, "Proof of authority : consensus model with identity at stake," 2017. [Online]. Available : https://medium.com/poa-network/proof-of-authority-consensus-model-with-identity-at-stake-d5bd15463256

[32] D. Ongaro and J. Ousterhout, "In search of an understandable consensus algorithm," in *2014 USENIX Annual Technical Conference (USENIXATC 14)*, 2014, pp. 305–319.

[33] F. Hofmann, S. Wurster, E. Ron, and M. Böhmecke-Schwafert, "The immutability concept of blockchains and benefits of early standardization," in *2017 ITU Kaleidoscope : Challenges for a Data-Driven Society (ITU K)*. IEEE, 2017, pp. 1–8.

[34] P. De Filippi, "The interplay between decentralization and privacy : the case of blockchain technologies," *Journal of Peer Production, Issue*, no. 7, 2016.

[35] T. Mitani and A. Otsuka, "Traceability in permissioned blockchain," *IEEE Access*, vol. 8, pp. 21 573–21 588, 2020.

[36] X. Li, P. Jiang, T. Chen, X. Luo, and Q. Wen, "A survey on the security of blockchain systems," *Future Generation Computer Systems*, vol. 107, pp. 841–853, 2020.

[37] R. C. Merkle, "A Digital Signature Based on a Conventional Encryption Function," in *Advances in Cryptology — CRYPTO '87*, C. Pomerance, Ed. Berlin, Heidelberg : Springer Berlin Heidelberg, 1988, pp. 369–378.

[38] H. Mayer, "Ecdsa security in bitcoin and ethereum : a research survey," *CoinFaabrik, June*, vol. 28, no. 126, p. 50, 2016.

[39] M. Dworkin, "Sha-3 standard : Permutation-based hash and extendable-output functions," 2015-08-04 2015.

[40] N. Szabo, "Formalizing and securing relationships on public networks," *First monday*, 1997.

[41] E. A. al., "Hyperledger fabric : a distributed operating system for permissioned blockchains," in *the Thirteenth EuroSys Conference on - EuroSys '18, Porto, Portugal*. Proceedings of, 2018, pp. 1–15.

[42] N. El Ioini and C. Pahl, "A Review of Distributed Ledger Technologies," in *On the Move to Meaningful Internet Systems. OTM 2018 Conferences*, H. Panetto, C. Debruyne, H. A. Proper, C. A. Ardagna, D. Roman, and R. Meersman, Eds. Cham : Springer International Publishing, 2018, pp. 277–288.

[43] Maker, "The Dai Stablecoin System," 2017. [Online]. Available : https://makerdao.com/whitepaper/DaiDec17WP.pdf

[44] K. Bhargavan, A. Delignat-Lavaud, C. Fournet, A. Gollamudi, G. Gonthier, N. Kobeissi, N. Kulatova, A. Rastogi, T. Sibut-Pinote, N. Swamy *et al.*, "Formal verification of smart contracts : Short paper," in *Proceedings of the 2016 ACM workshop on programming languages and analysis for security*, 2016, pp. 91–96.

[45] P. Tsankov, A. Dan, D. Drachsler-Cohen, A. Gervais, F. Buenzli, and M. Vechev, "Securify : Practical security analysis of smart contracts," in *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*, 2018, pp. 67–82.

[46] T. Dickerson, P. Gazzillo, M. Herlihy, and E. Koskinen, "Adding concurrency to smart contracts," *Distributed Computing*, pp. 1–17, 2019.

[47] Canadian Institute of Chartered Accountants, *Dictionnaire de la comptabilité et de la gestion financière, version 1.2*. Canadian Institute of Chartered Accountants, 2006.

[48] B. Magri, C. Matt, J. B. Nielsen, and D. Tschudi, "Afgjort – A Semi-Synchronous Finality Layer for Blockchains," *Self-published paper*, p. 44, 2019.

[49] M. Macdonald, L. Liu-Thorrold, and R. Julien, "The blockchain : a comparison of platforms and their uses beyond bitcoin," *Work. Pap*, pp. 1–18, 2017.

[50] P. Das, L. Eckey, T. Frassetto, D. Gens, K. Hostáková, P. Jauernig, S. Faust, and A.-R. Sadeghi, "Fastkitten : Practical smart contracts on bitcoin," in *28th {USENIX} Security Symposium ({USENIX} Security 19)*, 2019, pp. 801–818.

[51] R. Gupta, A. Shukla, and S. Tanwar, "Aayush : A smart contract-based telesurgery system for healthcare 4.0," in *2020 IEEE International Conference on Communications Workshops (ICC Workshops)*. IEEE, 2020, pp. 1–6.

[52] "Cryptocurrency Prices, Charts And Market Capitalizations." [Online]. Available : https://coinmarketcap.com

[53] P. Wegner, "Interoperability," *ACM Computing Surveys (CSUR)*, vol. 28, no. 1, pp. 285–287, 1996.

[54] A. Tolk and J. A. Muguira, "The levels of conceptual interoperability model," in *Proceedings of the 2003 fall simulation interoperability workshop*, vol. 7. Citeseer, 2003, pp. 1–11.

[55] S. Chiu, "Can Level of Information Systems Interoperability (LISI) Improve DoD C4I Systems' Interoperability ?" NAVAL POSTGRADUATE SCHOOL MONTEREY CA, Tech. Rep., 2001.

[56] N. Figay, "Interopérabilité Opérationnelle Continue," Mémoire d'Habilitation à Diriger des Recherches, Université de Lyon, Lyon, Jul. 2021.

[57] L. Richardson and S. Ruby, *RESTful web services*. " O'Reilly Media, Inc.", 2008.

[58] Twitter, "What's New with Twitter API v2." [Online]. Available : https://developer.twitter.com/en/docs/twitter-api/early-access

[59] E. Simperl, "Reusing ontologies on the semantic web : A feasibility study," *Data & Knowledge Engineering*, vol. 68, no. 10, pp. 905–925, 2009.

[60] D. L. McGuinness, F. Van Harmelen *et al.*, "Owl web ontology language overview," *W3C recommendation*, vol. 10, no. 10, p. 2004, 2004.

[61] H. Mokeddem and C. Desmoulins, "Intégration de raisonnements automatiques dans le système d'annotation memonote," in *23es Journées Francophones d'ingénierie des connaissances*, 2012, pp. 49–64.

[62] O. Iroju, A. Soriyan, I. Gambo, and J. Olaleke, "Interoperability in healthcare : benefits, challenges and resolutions," *International Journal of Innovation and Applied Studies*, vol. 3, no. 1, pp. 262–270, 2013.

[63] C. Meilicke, "Alignment incoherence in ontology matching," 2011.

[64] P. Livet, D. Phan, and L. Sanders, "Why do we need ontology for agent-based models ?" in *Complexity and artificial markets*. Springer, 2008, pp. 133–145.

[65] S. K. UntungRahardja and Q. EkaPurnamaHarahap, "Authenticity of a diploma using the blockchain approach," *International Journal*, vol. 9, no. 1.2, 2020.

[66] H. Dang, T. T. A. Dinh, D. Loghin, E.-C. Chang, Q. Lin, and B. C. Ooi, "Towards scaling blockchain systems via sharding," in *Proceedings of the 2019 international conference on management of data*, 2019, pp. 123–140.

[67] "BTC Relay," Sep. 2021, original-date : 2015-06-11T09 :14 :47Z. [Online]. Available : https://github.com/ethereum/btcrelay

[68] R. Belchior, "The State of Blockchain Interoperability in 2021," Mar. 2021. [Online]. Available : https://medium.com/coinmonks/the-state-of-blockchain-interoperability-in-2021-bcedaaa93ba5

[69] L. Besançon, P. Ghodous, J.-P. Gelas, and C. Ferreira da Silva, "Modelling of Decentralised Blockchain Applications Development," in *The 2020 International Conference on High Performance Computing & Simulation (HPCS 2020)*, Barcelone, Spain, Mar. 2021. [Online]. Available : https://hal.archives-ouvertes.fr/hal-03340842

[70] R. Belchior, A. Vasconcelos, S. Guerreiro, and M. Correia, "A survey on blockchain interoperability : Past, present, and future trends," *arXiv preprint arXiv :2005.14282*, 2020.

[71] L. Besançon, C. F. Da Silva, and P. Ghodous, "Towards blockchain interoperability : Improving video games data exchange," in *2019 IEEE International Conference on Blockchain and Cryptocurrency (ICBC)*. IEEE, 2019, pp. 81–85.

[72] A. Yahyavi and B. Kemme, "Peer-to-peer architectures for massively multiplayer online games : A survey," *ACM Computing Surveys (CSUR)*, vol. 46, no. 1, pp. 1–51, 2013.

[73] E. Christopoulou and S. Xinogalos, "Overview and comparative analysis of game engines for desktop and mobile devices," *International Journal of Serious Games*, vol. 4, no. 4, 2017.

[74] "Unreal Engine | The most powerful real-time 3D creation platform." [Online]. Available : https://www.unrealengine.com/en-US/

[75] U. Technologies, "Plateforme de développement en temps réel Unity | Moteur pour 3D, 2D, RV et RA." [Online]. Available : https://unity.com/fr

[76] U.-R. Hector and C.-L. Boris, "Blondie : Blockchain ontology with dynamic extensibility," *arXiv preprint arXiv :2008.09518*, 2020.

[77] J. Pfeffer, "EthOn — introducing semantic Ethereum," 2017. [Online]. Available : https://media.consensys.net/ethon-introducing-semantic-ethereum-15f1f0696986

[78] EverdreamSoft, "everdreamsoft/sandra." [Online]. Available : https://github.com/everdreamsoft/sandra

[79] everdreamsoft, "everdreamsoft/CrystalSpark-Cannon," May 2021, original-date : 2019-07-08T08 :57 :09Z. [Online]. Available : https://github.com/everdreamsoft/CrystalSpark-Cannon

[80] J. Parkkila, F. Radulovic, D. Garijo, M. Poveda-Villalón, J. Ikonen, J. Porras, and A. Gómez-Pérez, "An ontology for videogame interoperability," *Multimedia tools and applications*, vol. 76, no. 4, pp. 4981–5000, 2017.

[81] J. Zagal and A. Bruckman, "The game ontology project : Supporting learning while contributing authentically to game studies," *ICLS 2008*, 2008.

[82] C. Udokwu, H. Anyanka, and A. Norta, "Evaluation of approaches for designing and developing decentralized applications on blockchain," in *Proceedings of the 2020 4th International Conference on Algorithms, Computing and Systems*, ser. ICACS'20. New York, NY, USA : Association for Computing Machinery, 2020, p. 55–62. [Online]. Available : https://doi.org/10.1145/3423390.3426724

[83] M. Jurgelaitis, R. Butkienė, E. Vaičiukynas, V. Drungilas, and L. Čeponienė, "Modelling principles for blockchain-based implementation of business or scientific processes," in *CEUR workshop proceedings : IVUS 2019 international conference on information technologies : proceedings of the international conference on information technologies, Kaunas, Lithuania, April 25, 2019*, vol. 2470. CEUR-WS, 2019, pp. 43–47.

[84] N. Gull, M. Rashid, F. Azam, Y. Rasheed, and M. Waseem Anwar, "A block-chain oriented model driven framework for handling inconsistent requirements in global software development," in *2021 10th International Conference on Software and Computer Applications*, 2021, pp. 105–111.

[85] A. Mavridou and A. Laszka, "Designing secure ethereum smart contracts : A finite state machine based approach," 2017.

[86] S. A. White, "Introduction to BPMN," *Ibm Cooperation*, vol. 2, no. 0, 2004.

[87] T. Allweyer, *BPMN 2.0 : introduction to the standard for business process modeling*. BoD–Books on Demand, 2016.

[88] G. Decker and F. Puhlmann, "Extending BPMN for modeling complex choreographies," in *On the Move to Meaningful Internet Systems 2007 : CoopIS, DOA, ODBASE, GADA, and IS*, R. Meersman and Z. Tari, Eds. Berlin, Heidelberg : Springer Berlin Heidelberg, 2007, pp. 24–40.

[89] R. Banno and K. Shudo, "Simulating a blockchain network with simblock," in *2019 IEEE International Conference on Blockchain and Cryptocurrency (ICBC)*. IEEE, 2019, pp. 3–4.

[90] N. Kannengießer, M. Pfister, M. Greulich, S. Lins, and A. Sunyaev, "Bridges between islands : Cross-chain technology for distributed ledger technology," in *Proceedings of the 53rd Hawaii International Conference on System Sciences*, 2020.

[91] V. Buterin, "Chain interoperability," 2016.

[92] V. A. Siris, P. Nikander, S. Voulgaris, N. Fotiou, D. Lagutin, and G. C. Polyzos, "Interledger approaches," *IEEE Access*, vol. 7, pp. 89 948–89 966, 2019.

[93] K. Qin and A. Gervais, "An overview of blockchain scalability, interoperability and sustainability," *Hochschule Luzern Imperial College London Liquidity Network*, 2018.

[94] P. Frauenthaler, M. Sigwart, C. Spanring, M. Sober, and S. Schulte, "Eth relay : A cost-efficient relay for ethereum-based blockchains," in *2020 IEEE International Conference on Blockchain (Blockchain)*. IEEE, 2020, pp. 204–213.

[95] A. Zamyatin, Z. Avarikioti, D. Perez, and W. J. Knottenbelt, "Txchain : Efficient cryptocurrency light clients via contingent transaction aggregation," in *Data Privacy Management, Cryptocurrencies and Blockchain Technology*. Springer, 2020, pp. 269–286.

[96] S. Paavolainen and C. Carr, "Security properties of light clients on the ethereum blockchain," *IEEE Access*, vol. 8, pp. 124 339–124 358, 2020.

[97] E. Buchman, "Tendermint : Byzantine fault tolerance in the age of blockchains," Ph.D. dissertation, The University of Guelph, 2016.

[98] K. Chatterjee, A. K. Goharshady, and Y. Velner, "Quantitative analysis of smart contracts," in *European Symposium on Programming*. Springer, Cham, 2018, pp. 739–767.

[99] Y. Chen and C. Bellavitis, "Blockchain disruption and decentralized finance : The rise of decentralized business models," *Journal of Business Venturing Insights*, vol. 13, p. e00151, 2020.

[100] T. Oberstein, "EIP-665 : Add precompiled contract for Ed25519 signature verification." [Online]. Available : https://eips.ethereum.org/EIPS/eip-665

[101] M. H. Swende and N. Johnson, "EIP-191 : Signed Data Standard." [Online]. Available : https://eips.ethereum.org/EIPS/eip-191

[102] F. Vogelsteller and V. Buterin, "Erc-20 token standard," 2015. [Online]. Available : https://github.com/ethereum/EIPs/blob/master/EIPS/eip-20.md

[103] [Online]. Available : https://blockchain.ieee.org/standards

[104] [Online]. Available : https://standards.ieee.org

[105] [Online]. Available : https://standards.ieee.org/project/2418_1.html

[106] A. Regenscheid and D. Yaga, "Blockchain and distributed ledger technologies : Opportunities, challenges and future work," 2017. [Online]. Available : https://csrc.nist.gov/CSRC/media/Presentations/NIST-Block-Chain-Research-Project/images-media/ar-dy-blockchain-combined.pdf

[107] [Online]. Available : https://entethalliance.org

[108] E. Hildenbrandt, M. Saxena, N. Rodrigues, X. Zhu, P. Daian, D. Guth, B. Moore, D. Park, Y. Zhang, A. Stefanescu *et al.*, "Kevm : A complete formal semantics of the ethereum virtual machine," in *2018 IEEE 31st Computer Security Foundations Symposium (CSF)*. IEEE, 2018, pp. 204–217.

[109] "Introduction | MetaMask Docs." [Online]. Available : https://docs.metamask.io/guide/

[110] "Getting started with the Polygon SDK | Matic Network | Documentation." [Online]. Available : https://docs.matic.network//docs/develop/ethereum-matic/pos/using-sdk/getting-started

[111] "HTTP API." [Online]. Available : https://docs.ipfs.io/reference/http/api/

[112] Matrix, "Matrix Specification." [Online]. Available : https://matrix.org/docs/spec

[113] "ethereum/devp2p." [Online]. Available : https://github.com/ethereum/devp2p

[114] T. Cerny, M. J. Donahoo, and M. Trnka, "Contextual Understanding of Microservice Architecture : Current and Future Directions," *SIGAPP Appl. Comput. Rev.*, vol. 17, no. 4, pp. 29–45, Jan. 2018. [Online]. Available : http://doi.acm.org/10.1145/3183628.3183631

[115] W. Cai, Z. Wang, J. B. Ernst, Z. Hong, C. Feng, and V. C. M. Leung, "Decentralized Applications : The Blockchain-Empowered Software System," *IEEE Access*, vol. 6, pp. 53 019–53 033, 2018.

[116] Hoard, "True ownership," 2019. [Online]. Available : https://hoard.exchange/files/hoard-whitepaper.pdf

[117] H.-W. Kim and Y.-S. Jeong, "Secure authentication-management human-centric scheme for trusting personal resource information on mobile cloud computing with blockchain," *Human-centric Computing and Information Sciences*, vol. 8, no. 1, pp. 1–13, 2018.

[118] A. Kryukov and A. Demichev, "Decentralized data storages : Technologies of construction," *Programming and Computer Software*, vol. 44, no. 5, pp. 303–315, 2018.

[119] M. Sabt, M. Achemlal, and A. Bouabdallah, "Trusted execution environment : what it is, and what it is not," in *2015 IEEE Trustcom/BigDataSE/ISPA*, vol. 1. IEEE, 2015, pp. 57–64.

[120] R. Viswanathan, D. Dasgupta, and S. R. Govindaswamy, "Blockchain Solution Reference Architecture (BSRA)," *IBM Journal of Research and Development*, vol. 63, no. 2/3, pp. 1 :1–1 :12, Mar. 2019.

[121] G. S. Ramachandran and B. Krishnamachari, "A reference architecture for blockchain-based peer-to-peer iot applications," *arXiv preprint arXiv :1905.10643*, 2019.

[122] W. Warren and A. Bandeali, "0x : An open protocol for decentralized exchange on the ethereum blockchain," *URl : https ://github.com/0xProject/whitepaper*, pp. 04–18, 2017.

[123] B. DiFrancesco, "Validium And The Layer 2 Two-By-Two — Issue No. 99." [Online]. Available : https://www.buildblockchain.tech/newsletter/issues/no-99-validium-and-the-layer-2-two-by-two

[124] F. Technologies, "Blockchain solutions for gaming. whitepaper v2 draft." [Online]. Available : https://funfair.io/wp-content/uploads/FunFair-Commercial-White-Paper-v2-draft.pdf

[125] J. Poon and T. Dryja, "The bitcoin lightning network : Scalable off-chain instant payments," 2016. [Online]. Available : https://lightning.network/lightning-network-paper.pdf

[126] J. Coleman, L. Horne, and L. Xuanji, "Counterfactual : Generalized state channels." [Online]. Available : https://l4.ventures/papers/statechannels.pdf

[127] V. Buterin, "An Incomplete Guide to Rollups." [Online]. Available : https://vitalik.ca/general/2021/01/05/rollup.html

[128] "Loopring Protocol Design," Sep. 2021, original-date : 2018-12-29T00 :39 :38Z. [Online]. Available : https://github.com/Loopring/protocols/blob/fe1f69c3769f74a089bf9180456aa46b6b13e180/packages/loopring_v3/DESIGN.md

[129] "Overview | zkSync : secure, scalable crypto payments." [Online]. Available : https://zksync.io/faq/intro.html#introduction

[130] "Inside Arbitrum · Offchain Labs Dev Center." [Online]. Available : https://developer.offchainlabs.com/

[131] "Introduction | StarkEx V3." [Online]. Available : https://docs.starkware.co/starkex-v3/

[132] "DeversiFi." [Online]. Available : https://docs.deversifi.com/articles#HowDeversiFiWorks

[133] A. B. al., "Enabling blockchain innovations with pegged sidechains," 2014. [Online]. Available : http://kevinriggen.com/files/sidechains.pdf

[134] J. Poon and V. Buterin, "Plasma : Scalable autonomous smart contracts," 2017. [Online]. Available : https://plasma.io/plasma.pdf

[135] L. Network, "Loom network - the next-generation blockchain application platform for ethereum." [Online]. Available : https://loomx.io

[136] Polygon, "Polygon - ethereum's internet of blockchains," Feb 2021. [Online]. Available : https://polygon.technology/lightpaper-polygon.pdf

[137] J. Kanani, Sandeep Nailwal, and A. Arjun, "Matic Whitepaper," Sep. 2021, original-date : 2018-06-12T06 :04 :51Z. [Online]. Available : https://github.com/maticnetwork/whitepaper

[138] G. Wood, "Polkadot : Vision for a heterogeneous multi-chain framework," *White Paper*, vol. 21, 2016.

[139] "Polygon | Ethereum's Internet of Blockchains." [Online]. Available : https://polygon.technology/

[140] J. Kwon and E. Buchman, "Cosmos whitepaper," 2019.

[141] "Harmony – Scaling Ethereum Applications and Cross-Chain Finance." [Online]. Available : https://www.harmony.one/

[142] J. Benet, "Ipfs-content addressed, versioned, p2p file system," *arXiv preprint arXiv :1407.3561*, 2014.

[143] M. Zichichi, S. Ferretti, and G. D'Angelo, "On the efficiency of decentralized file storage for personal information management systems," 2020.

[144] Filecoin, "A decentralized storage network for humanity's most important information." [Online]. Available : https://filecoin.io/

[145] S. Wilkinson, T. Boshevski, J. Brandoff, and V. Buterin, "Storj a peer-to-peer cloud storage network," *Storj Labs, Inc.*, 2014.

[146] K. Chatterjee, A. K. Goharshady, and A. Pourdamghani, "Probabilistic smart contracts : Secure randomness on the blockchain," in *2019 IEEE International Conference on Blockchain and Cryptocurrency (ICBC)*. IEEE, 2019, pp. 403–412.

[147] iExec, "Blockchain-based decentralized cloud computing | whitepaper." [Online]. Available : https://iex.ec/wp-content/uploads/pdf/iExec-WPv3.0-English.pdf

[148] J. Teutsch and C. Reitwießner, "A scalable verification solution for blockchains," *arXiv preprint arXiv :1908.04756*, 2019.

[149] "Cryptocurrency Exchange EtherDelta Hacked in DNS Hijacking Scheme," Dec. 2017. [Online]. Available : https://www.ccn.com/cryptocurrency-exchange-etherdelta-hacked-in-dns-hijacking-scheme/

[150] J. Benet, "Ipfs-content addressed, versioned, p2p file system," *arXiv preprint arXiv :1407.3561*, 2014.

[151] "Tug Of War, An Unstoppable Game." [Online]. Available : http://tugofwar.io

[152] R. Balzer and N. Goldman, "Principles of good software specification and their implications for specification languages," in *Proceedings of the May 4-7, 1981, National Computer Conference*, ser. AFIPS '81. New York, NY, USA : ACM, 1981, pp. 393–400. [Online]. Available : http://doi.acm.org/10.1145/1500412.1500468

[153] J. Zahnentferner, "Chimeric ledgers : Translating and unifying utxo-based and account-based cryptocurrencies." *IACR Cryptology ePrint Archive*, vol. 2018, p. 262, 2018.

[154] X. Xu, I. Weber, and M. Staples, "Blockchain patterns," in *Architecture for Blockchain Applications*. Springer, 2019, pp. 113–148.

[155] A. Ahmed Nacer, "Composition sure d'API fondée sur des contrats," Thèse de Doctorat, Université de Lorraine, 2021.

[156] "Mintable.app - Home." [Online]. Available : https://mintable.app/

[157] OpenSea, "OpenSea, the largest NFT marketplace." [Online]. Available : https://opensea.io/

[158] C. Jentzsch, "Decentralized autonomous organization to automate governance," *White paper, November*, 2016.

[159] B. Liu, P. Szalachowski, and J. Zhou, "A First Look into DeFi Oracles," *arXiv :2005.04377 [cs]*, Jun. 2021, arXiv : 2005.04377. [Online]. Available : http://arxiv.org/abs/2005.04377

[160] G. M. Hastig and M. S. Sodhi, "Blockchain for supply chain traceability : Business requirements and critical success factors," *Production and Operations Management*, vol. 29, no. 4, pp. 935–954, 2020.

[161] V. Ojha, "GitHub - IBM/BlockchainDevelopmentDesign-Patterns." [Online]. Available : https://github.com/IBM/BlockchainDevelopmentDesignPatterns

[162] I. Horrocks, P. F. Patel-Schneider, H. Boley, S. Tabet, B. Grosof, M. Dean *et al.*, "Swrl : A semantic web rule language combining owl and ruleml," *W3C Member submission*, vol. 21, no. 79, pp. 1–31, 2004.

[163] J. Nguyen, J. Geyer, T. Farrenkopf, and M. Guckert, "Aided OWL Notation (AOWLN) : Conceptual Modelling and Visualisation of Advanced SWRL Rules :," in *Proceedings of the 10th International Joint Conference on Knowledge Discovery, Knowledge Engineering and Knowledge Management*. Seville, Spain : SCITEPRESS - Science and Technology Publications, 2018, pp. 175–182. [Online]. Available : http://www.scitepress.org/DigitalLibrary/Link.aspx?doi=10.5220/0006917701750182

[164] "SWRLTemporalBuiltInsBasic · protegeproject/swrlapi Wiki." [Online]. Available : https://github.com/protegeproject/swrlapi

[165] M. Proctor, "Drools : a rule engine for complex event processing," in *International Symposium on Applications of Graph Transformations with Industrial Relevance*. Springer, 2011, pp. 2–2.

[166] "Home · protegeproject/swrlapi Wiki." [Online]. Available : https://github.com/protegeproject/swrlapi

[167] O. Choudhury, M. Dhuliawala, N. Fay, N. Rudolph, I. Sylla, N. Fairoza, D. Gruen, and A. Das, "Auto-translation of regulatory documents into smart contracts," *IEEE Blockchain Initiative,(September)*, pp. 1–5, 2018.

[168] X. Lou, K. Hwang *et al.*, "Adaptive content poisoning to prevent illegal file distribution in p2p networks," *IEEE Trans. Computers*, 2006.

[169] BGA, "Blockchain Game Alliance – Helping bring the power of Blockchain to the gaming industry." [Online]. Available : https://www.blockchaingamealliance.org

[170] S. Heintz and E. L.-C. Law, "The game genre map : A revised game classification," in *Proceedings of the 2015 Annual Symposium on Computer-Human Interaction in Play*, 2015, pp. 175–184.

[171] D. J. Barnes and J. Hernandez-Castro, "On the limits of engine analysis for cheating detection in chess," *Computers & Security*, vol. 48, pp. 58–73, 2015.

Annexe A

Axiomes inférés par Drools

Cette annexe est un complément à la section 5.1.3 et présente la liste exhaustive des axiomes relatifs aux propriétés de données ou d'objets inférés par le moteur d'inférence Drools.

```
sbo:_DBA1 sbo:NbMbPerUser "0.0"^^xsd:double
sbo:_IPFS sbo:LatencyReads "7.0"^^xsd:double
sbo:_DBA3 owl:topObjectProperty sbo:_MaticMainnet
<http://ethereum.ethstats.io/EthereumFoundationMainChain> ethon:containsBlock <http
    ://ethereum.ethstats.io/Block_1920000>
sbo:_CentralizedStorage sbo:LatencyReads "0.1"^^xsd:double
<http://ethereum.ethstats.io/Block_2463000> owl:topDataProperty 2463000
<http://ethereum.ethstats.io/Block_2675000> ethon:NetworkObjectProperty ethon:
    ProtocolVariant_SpuriousDragon
sbo:_DBA3 sbo:NbMbDevPerUser "1000.0"^^xsd:double
ethon:ProtocolVariant_SpuriousDragon ethon:NetworkObjectProperty ethon:
    ProtocolVariant_Byzantium
ethon:ProtocolVariant_Homestead ethon:NetworkObjectProperty ethon:
    ProtocolVariant_DAO
sbo:_MaticLayerTwo sbo:LatencyWrites "2.0"^^xsd:double
<http://ethereum.ethstats.io/EthereumFoundationMainChain> ethon:EthOnObjectProperty
     <http://ethereum.ethstats.io/Block_1150000>
<http://ethereum.ethstats.io/EthereumFoundationMainChain> ethon:
    NetworkObjectProperty <http://ethereum.ethstats.io/Block_4370000>
sbo:_DBA1 sbo:Latency "0.0"^^xsd:double
sbo:_DBA1 sbo:Latency "0.2"^^xsd:double
sbo:_DBA1 sbo:Latency "0.1"^^xsd:double
<http://ethereum.ethstats.io/EthereumFoundationMainChain> ethon:EthOnObjectProperty
     <http://ethereum.ethstats.io/Block_2675000>
sbo:_DBA2 sbo:NbMb "1.0"^^xsd:double
<http://ethereum.ethstats.io/Block_2463000> ethon:EthOnDataProperty 2463000
sbo:_MaticCheckpoint1 gbo:Contains sbo:_MaticTransaction1
sbo:_DBA2 owl:topDataProperty "10.0"^^xsd:double
<http://ethereum.ethstats.io/Block_0> ethon:NetworkObjectProperty ethon:
    ProtocolVariant_Frontier
sbo:_MaticLayerTwo owl:topDataProperty "0.001"^^xsd:double
sbo:_MaticLayerTwo owl:topDataProperty "0.0"^^xsd:double
sbo:_DBA3 sbo:NbTxPerUser "0.0"^^xsd:double
sbo:_DBA1 owl:topObjectProperty sbo:_Unity
sbo:_DBA2 sbo:NbTx "10.0"^^xsd:double
ethon:ProtocolVariant_IceAge ethon:NetworkObjectProperty ethon:
    ProtocolVariant_Homestead
sbo:_NullStorage owl:topDataProperty "0.0"^^xsd:double
sbo:_MaticTransaction3 owl:topObjectProperty sbo:_MaticCheckpoint2
sbo:_DBA2 owl:topObjectProperty sbo:_CentralizedStorage
sbo:_DBA1 sbo:NbTxPerUser "10.0"^^xsd:double
```

```
ethon:ProtocolVariant_Frontier ethon:NetworkObjectProperty ethon:ProtocolVariant_IceAge
<http://ethereum.ethstats.io/EthereumFoundationMainChain> ethon:NetworkObjectProperty <http://ethereum.ethstats.io/Block_200000>
sbo:_DBA3 owl:topObjectProperty sbo:_NullLayerTwo
<http://ethereum.ethstats.io/Block_2463000> ethon:NetworkObjectProperty ethon:ProtocolVariant_TangerineWhistle
sbo:_DBA2 sbo:NbMb "0.0"^^xsd:double
sbo:_MaticMainnet sbo:Latency "2.0"^^xsd:double
sbo:_DBA2 owl:topDataProperty "1000.0"^^xsd:double
<http://ethereum.ethstats.io/Block_1150000> gbo:HasHash 1150000
sbo:_DBA2 sbo:NbTxPerUser "0.0"^^xsd:double
sbo:_DBA1 sbo:NbTxDevPerUser "2.0"^^xsd:double
<http://ethereum.ethstats.io/EthereumFoundationMainChain> ethon:NetworkObjectProperty <http://ethereum.ethstats.io/Block_2463000>
ethon:ProtocolVariant_IceAge ethon:EthOnObjectProperty ethon:ProtocolVariant_Homestead
sbo:_MaticTransaction2 gbo:IncludedIn sbo:_MaticCheckpoint2
sbo:_EthereumMainnet sbo:Cost "1.0"^^xsd:double
<http://ethereum.ethstats.io/EthereumFoundationMainChain> ethon:NetworkObjectProperty <http://ethereum.ethstats.io/Block_2675000>
sbo:_DBA2 owl:topObjectProperty sbo:_Unity
sbo:_MaticCheckpoint1 owl:topObjectProperty sbo:_MaticTransaction1
<http://ethereum.ethstats.io/EthereumFoundationMainChain> ethon:containsBlock <http://ethereum.ethstats.io/Block_2675000>
ethon:ProtocolVariant_TangerineWhistle ethon:EthOnObjectProperty ethon:ProtocolVariant_SpuriousDragon
sbo:_DBA2 sbo:Cost "2.0"^^xsd:double
sbo:_DBA3 sbo:NbTx "0.0"^^xsd:double
<http://ethereum.ethstats.io/Block_4370000> ethon:NetworkObjectProperty ethon:ProtocolVariant_Byzantium
 SameIndividual: ethon:ProtocolVariant_SpuriousDragon
<http://ethereum.ethstats.io/EthereumFoundationMainChain> ethon:EthOnObjectProperty <http://ethereum.ethstats.io/Block_1920000>
sbo:_DBA1 sbo:NbMb "0.0"^^xsd:double
ethon:ProtocolVariant_TangerineWhistle ethon:hasFork ethon:ProtocolVariant_SpuriousDragon
ethon:ProtocolVariant_IceAge ethon:hasFork ethon:ProtocolVariant_Homestead
<http://ethereum.ethstats.io/EthereumFoundationMainChain> ethon:containsBlock <http://ethereum.ethstats.io/Block_1150000>
sbo:_DBA1 sbo:NbMbDevPerUser "0.0"^^xsd:double
sbo:_MaticTransaction1 owl:topDataProperty "2021-10-18T18:20:00"^^xsd:dateTime
sbo:_P2PStorage sbo:Cost "0.0"^^xsd:double
sbo:_EthereumMainnet sbo:LatencyWrites "14.0"^^xsd:double
sbo:_MaticLayerTwo sbo:LatencyReads "0.0"^^xsd:double
sbo:_DBA3 owl:topDataProperty "0.2"^^xsd:double
sbo:_DBA3 owl:topDataProperty "0.01"^^xsd:double
sbo:_DBA3 owl:topDataProperty "0.002"^^xsd:double
sbo:_DBA3 owl:topDataProperty "0.1"^^xsd:double
sbo:_DBA3 owl:topDataProperty "0.0"^^xsd:double
ethon:ProtocolVariant_Frontier ethon:hasFork ethon:ProtocolVariant_IceAge
ethon:ProtocolVariant_DAO ethon:hasFork ethon:ProtocolVariant_TangerineWhistle
ethon:ProtocolVariant_Frontier ethon:EthOnObjectProperty ethon:ProtocolVariant_IceAge
sbo:_DBA3 sbo:NbTx "10.0"^^xsd:double
sbo:_DBA2 sbo:LatencyWrites "0.2"^^xsd:double
sbo:_DBA1 sbo:NbTxDevPerUser "0.0"^^xsd:double
sbo:_DBA2 sbo:NbTxPerUser "10.0"^^xsd:double
sbo:_IPFS owl:topDataProperty "7.0"^^xsd:double
<http://ethereum.ethstats.io/EthereumFoundationMainChain> ethon:NetworkObjectProperty <http://ethereum.ethstats.io/Block_1150000>
<http://ethereum.ethstats.io/EthereumFoundationMainChain> ethon:EthOnObjectProperty <http://ethereum.ethstats.io/Block_200000>
sbo:_MaticMainnet sbo:Cost "0.001"^^xsd:double
<http://ethereum.ethstats.io/EthereumFoundationMainChain> ethon:EthOnObjectProperty <http://ethereum.ethstats.io/Block_4370000>
<http://ethereum.ethstats.io/Block_1150000> ethon:EthOnObjectProperty ethon:ProtocolVariant_Homestead
```

```
sbo:_DBA1 sbo:LatencyReads "0.1"^^xsd:double
sbo:_DBA1 sbo:LatencyReads "0.0"^^xsd:double
sbo:_DBA3 owl:topDataProperty "1000.0"^^xsd:double
 SameIndividual: ethon:ProtocolVariant_TangerineWhistle
<http://ethereum.ethstats.io/Block_1920000> ethon:EthOnDataProperty 1920000
sbo:_DBA3 sbo:LatencyWrites "2.0"^^xsd:double
sbo:_CentralizedStorage sbo:LatencyWrites "0.2"^^xsd:double
<http://ethereum.ethstats.io/EthereumFoundationMainChain> ethon:containsBlock <http
    ://ethereum.ethstats.io/Block_4370000>
sbo:_DBA3 sbo:NbMb "1000.0"^^xsd:double
sbo:_CentralizedStorage owl:topDataProperty "0.2"^^xsd:double
sbo:_CentralizedStorage owl:topDataProperty "0.1"^^xsd:double
sbo:_CentralizedStorage owl:topDataProperty "0.0"^^xsd:double
sbo:_DBA3 owl:topDataProperty "10.0"^^xsd:double
sbo:_EthereumMainnet owl:topDataProperty "5000.0"^^xsd:double
sbo:_MaticLayerTwo owl:topDataProperty 5
 SameIndividual: ethon:ProtocolVariant_Homestead
sbo:_DBA2 sbo:NbMbPerUser "1.0"^^xsd:double
sbo:_MaticMainnet owl:topDataProperty "5.0"^^xsd:double
sbo:_DBA2 sbo:NbMbDevPerUser "0.0"^^xsd:double
sbo:_DBA2 sbo:LatencyWrites "2.0"^^xsd:double
sbo:_DBA1 sbo:NbTx "2.0"^^xsd:double
sbo:_DBA1 owl:topObjectProperty sbo:_CentralizedStorage
sbo:_MaticCheckpoint2 owl:topObjectProperty sbo:_MaticTransaction2
sbo:_DBA1 sbo:Latency "14.0"^^xsd:double
sbo:_P2PStorage sbo:Latency "0.5"^^xsd:double
sbo:_DBA2 owl:topDataProperty "0.1"^^xsd:double
sbo:_DBA2 owl:topDataProperty "0.0"^^xsd:double
sbo:_DBA2 owl:topDataProperty "0.01"^^xsd:double
sbo:_DBA2 owl:topDataProperty "0.2"^^xsd:double
sbo:_P2PStorage sbo:LatencyReads "0.5"^^xsd:double
sbo:_MaticLayerTwo owl:topDataProperty "2.0"^^xsd:double
<http://ethereum.ethstats.io/Block_200000> ethon:EthOnObjectProperty ethon:
    ProtocolVariant_IceAge
sbo:_DBA1 sbo:NbTxPerUser "0.0"^^xsd:double
sbo:_MaticMainnet sbo:Latency "0.0"^^xsd:double
sbo:_DBA3 sbo:LatencyWrites "0.2"^^xsd:double
sbo:_DBA2 sbo:NbMbPerUser "0.0"^^xsd:double
sbo:_MaticTransaction3 gbo:IncludedIn sbo:_MaticCheckpoint2
<http://ethereum.ethstats.io/Block_200000> ethon:EthOnDataProperty 200000
sbo:_DBA3 sbo:NbTxPerUser "10.0"^^xsd:double
sbo:_MaticCheckpoint2 owl:topObjectProperty sbo:_MaticTransaction3
sbo:_DBA3 owl:topObjectProperty sbo:_Unity
sbo:_DBA1 owl:topDataProperty "2.0"^^xsd:double
 SameIndividual: ethon:ProtocolVariant_Frontier
sbo:_DBA2 sbo:NbTx "0.0"^^xsd:double
sbo:_MaticTransaction3 owl:topDataProperty "2021-10-18T18:50:00"^^xsd:dateTime
sbo:_IPFS sbo:LatencyWrites "7.0"^^xsd:double
sbo:_EthereumMainnet owl:topDataProperty "14.0"^^xsd:double
sbo:_DBA1 sbo:Cost "10.0"^^xsd:double
<http://ethereum.ethstats.io/Block_2675000> ethon:EthOnObjectProperty ethon:
    ProtocolVariant_SpuriousDragon
sbo:_MaticMainnet sbo:LatencyReads "0.0"^^xsd:double
sbo:_DBA1 owl:topObjectProperty sbo:_EthereumMainnet
<http://ethereum.ethstats.io/Block_1920000> gbo:HasHash 1920000
sbo:_DBA3 sbo:NbTxDevPerUser "2.0"^^xsd:double
<http://ethereum.ethstats.io/Block_1150000> owl:topDataProperty 1150000
<http://ethereum.ethstats.io/Block_4370000> owl:topDataProperty 4370000
sbo:_EthereumMainnet owl:topDataProperty "1.0"^^xsd:double
sbo:_DBA2 sbo:NbMb "1000.0"^^xsd:double
<http://ethereum.ethstats.io/Block_1150000> ethon:NetworkObjectProperty ethon:
    ProtocolVariant_Homestead
sbo:_DBA2 owl:topDataProperty "2.0"^^xsd:double
sbo:_DBA1 sbo:NbTx "0.0"^^xsd:double
sbo:_MaticMainnet owl:topDataProperty "2.0"^^xsd:double
sbo:_DBA3 sbo:NbMbDevPerUser "0.0"^^xsd:double
sbo:_DBA2 sbo:NbTx "2.0"^^xsd:double
ethon:ProtocolVariant_DAO ethon:NetworkObjectProperty ethon:
```

```
      ProtocolVariant_TangerineWhistle
sbo:_DBA2 owl:topObjectProperty sbo:_MaticLayerTwo
sbo:_MaticLayerTwo sbo:Latency "2.0"^^xsd:double
sbo:_EthereumMainnet sbo:Cost "5000.0"^^xsd:double
sbo:_EthereumMainnet owl:topDataProperty "0.0"^^xsd:double
<http://ethereum.ethstats.io/Block_1920000> ethon:EthOnObjectProperty ethon:
      ProtocolVariant_DAO
sbo:_DBA1 owl:topDataProperty "0.2"^^xsd:double
sbo:_DBA1 owl:topDataProperty "0.1"^^xsd:double
sbo:_DBA2 owl:topDataProperty "1.0"^^xsd:double
sbo:_DBA1 owl:topDataProperty "0.0"^^xsd:double
<http://ethereum.ethstats.io/Block_4370000> ethon:EthOnDataProperty 4370000
sbo:_MaticCheckpoint1 owl:topDataProperty "2021-10-18T18:30:00"^^xsd:dateTime
sbo:_DBA2 sbo:LatencyReads "0.0"^^xsd:double
sbo:_DBA2 sbo:LatencyReads "0.1"^^xsd:double
sbo:_DBA1 owl:topDataProperty "14.0"^^xsd:double
<http://ethereum.ethstats.io/Block_2463000> gbo:HasHash 2463000
sbo:_DBA3 sbo:NbMbPerUser "0.0"^^xsd:double
sbo:_MaticTransaction1 owl:topObjectProperty sbo:_MaticCheckpoint1
 SameIndividual: ethon:ProtocolVariant_DAO
sbo:_IPFS owl:topDataProperty "0.0"^^xsd:double
<http://ethereum.ethstats.io/EthereumFoundationMainChain> ethon:
      NetworkObjectProperty <http://ethereum.ethstats.io/Block_0>
<http://ethereum.ethstats.io/Block_2675000> ethon:EthOnDataProperty 2675000
sbo:_DBA1 sbo:LatencyWrites "0.2"^^xsd:double
<http://ethereum.ethstats.io/Block_200000> owl:topDataProperty 200000
sbo:_DBA2 owl:topObjectProperty sbo:_EthereumMainnet
<http://ethereum.ethstats.io/EthereumFoundationMainChain> ethon:containsBlock <http
      ://ethereum.ethstats.io/Block_200000>
sbo:_NullLayerTwo sbo:Cost "0.0"^^xsd:double
<http://ethereum.ethstats.io/Block_200000> ethon:BlockDataProperty 200000
sbo:_DBA3 sbo:NbMbPerUser "1.0"^^xsd:double
sbo:_MaticMainnet owl:topDataProperty "0.0"^^xsd:double
sbo:_MaticMainnet owl:topDataProperty "0.001"^^xsd:double
<http://ethereum.ethstats.io/Block_1150000> ethon:EthOnDataProperty 1150000
sbo:_DBA2 sbo:NbTxDevPerUser "0.0"^^xsd:double
sbo:_DBA3 sbo:Cost "0.002"^^xsd:double
sbo:_DBA3 sbo:Cost "0.01"^^xsd:double
sbo:_NullLayerTwo owl:topDataProperty "0.0"^^xsd:double
ethon:ProtocolVariant_DAO ethon:EthOnObjectProperty ethon:
      ProtocolVariant_TangerineWhistle
<http://ethereum.ethstats.io/Block_2463000> ethon:BlockDataProperty 2463000
sbo:_P2PStorage owl:topDataProperty "0.5"^^xsd:double
sbo:_DBA1 owl:topObjectProperty sbo:_NullLayerTwo
sbo:_P2PStorage owl:topDataProperty "0.0"^^xsd:double
<http://ethereum.ethstats.io/Block_0> ethon:EthOnDataProperty 0
sbo:_MaticLayerTwo sbo:Latency "0.0"^^xsd:double
<http://ethereum.ethstats.io/Block_2463000> ethon:EthOnObjectProperty ethon:
      ProtocolVariant_TangerineWhistle
sbo:_IPFS sbo:Latency "7.0"^^xsd:double
sbo:_EthereumMainnet sbo:Latency "14.0"^^xsd:double
<http://ethereum.ethstats.io/Block_4370000> ethon:BlockDataProperty 4370000
sbo:_DBA3 owl:topObjectProperty sbo:_CentralizedStorage
sbo:_MaticTransaction2 owl:topDataProperty "2021-10-18T18:45:00"^^xsd:dateTime
sbo:_DBA2 sbo:Cost "0.01"^^xsd:double
<http://ethereum.ethstats.io/EthereumFoundationMainChain> ethon:EthOnObjectProperty
       <http://ethereum.ethstats.io/Block_0>
sbo:_DBA3 sbo:NbTx "2.0"^^xsd:double
sbo:_DBA1 sbo:LatencyWrites "14.0"^^xsd:double
<http://ethereum.ethstats.io/Block_200000> gbo:HasHash 200000
<http://ethereum.ethstats.io/EthereumFoundationMainChain> ethon:
      NetworkObjectProperty <http://ethereum.ethstats.io/Block_1920000>
sbo:_DBA3 owl:topDataProperty "1.0"^^xsd:double
<http://ethereum.ethstats.io/Block_0> ethon:EthOnObjectProperty ethon:
      ProtocolVariant_Frontier
sbo:_CentralizedStorage sbo:Latency "0.1"^^xsd:double
sbo:_CentralizedStorage sbo:Latency "0.2"^^xsd:double
<http://ethereum.ethstats.io/Block_0> ethon:BlockDataProperty 0
```

```
sbo:_DBA3 sbo:NbMb "0.0"^^xsd:double
sbo:_MaticCheckpoint2 gbo:Contains sbo:_MaticTransaction2
ethon:ProtocolVariant_TangerineWhistle ethon:NetworkObjectProperty ethon:
    ProtocolVariant_SpuriousDragon
<http://ethereum.ethstats.io/EthereumFoundationMainChain> ethon:EthOnObjectProperty
     <http://ethereum.ethstats.io/Block_2463000>
sbo:_IPFS sbo:Cost "0.0"^^xsd:double
ethon:ProtocolVariant_Homestead ethon:hasFork ethon:ProtocolVariant_DAO
sbo:_MaticCheckpoint2 owl:topDataProperty "2021-10-18T19:00:00"^^xsd:dateTime
sbo:_DBA3 sbo:Latency "2.0"^^xsd:double
sbo:_P2PStorage sbo:LatencyWrites "0.5"^^xsd:double
<http://ethereum.ethstats.io/Block_0> gbo:HasHash 0
ethon:ProtocolVariant_SpuriousDragon ethon:EthOnObjectProperty ethon:
    ProtocolVariant_Byzantium
sbo:_DBA2 sbo:NbTxDevPerUser "2.0"^^xsd:double
<http://ethereum.ethstats.io/Block_4370000> ethon:EthOnObjectProperty ethon:
    ProtocolVariant_Byzantium
sbo:_NullStorage sbo:Cost "0.0"^^xsd:double
sbo:_MaticMainnet sbo:LatencyWrites "2.0"^^xsd:double
sbo:_DBA3 owl:topDataProperty "2.0"^^xsd:double
<http://ethereum.ethstats.io/Block_1920000> owl:topDataProperty 1920000
sbo:_CentralizedStorage sbo:Cost "0.0"^^xsd:double
sbo:_MaticCheckpoint2 gbo:Contains sbo:_MaticTransaction3
sbo:_DBA3 sbo:NbMb "1.0"^^xsd:double
sbo:_MaticLayerTwo sbo:Cost 5
sbo:_DBA2 sbo:Latency "0.1"^^xsd:double
<http://ethereum.ethstats.io/Block_2675000> gbo:HasHash 2675000
sbo:_DBA2 sbo:Latency "0.2"^^xsd:double
sbo:_DBA2 sbo:Latency "0.0"^^xsd:double
<http://ethereum.ethstats.io/EthereumFoundationMainChain> ethon:containsBlock <http
    ://ethereum.ethstats.io/Block_0>
ethon:ProtocolVariant_SpuriousDragon ethon:hasFork ethon:ProtocolVariant_Byzantium
sbo:_DBA1 sbo:NbTx "10.0"^^xsd:double
sbo:_MaticLayerTwo sbo:Cost "0.001"^^xsd:double
<http://ethereum.ethstats.io/Block_2675000> ethon:BlockDataProperty 2675000
sbo:_MaticTransaction2 owl:topObjectProperty sbo:_MaticCheckpoint2
<http://ethereum.ethstats.io/Block_2675000> owl:topDataProperty 2675000
<http://ethereum.ethstats.io/EthereumFoundationMainChain> ethon:containsBlock <http
    ://ethereum.ethstats.io/Block_2463000>
<http://ethereum.ethstats.io/Block_200000> ethon:NetworkObjectProperty ethon:
    ProtocolVariant_IceAge
ethon:ProtocolVariant_Homestead ethon:EthOnObjectProperty ethon:ProtocolVariant_DAO

sbo:_EthereumMainnet sbo:Latency "0.0"^^xsd:double
sbo:_MaticTransaction1 gbo:IncludedIn sbo:_MaticCheckpoint1
sbo:_DBA3 sbo:LatencyReads "0.0"^^xsd:double
sbo:_DBA3 sbo:LatencyReads "0.1"^^xsd:double
sbo:_EthereumMainnet sbo:LatencyReads "0.0"^^xsd:double
<http://ethereum.ethstats.io/Block_4370000> gbo:HasHash 4370000
sbo:_DBA3 sbo:Latency "0.1"^^xsd:double
sbo:_DBA3 sbo:Latency "0.2"^^xsd:double
sbo:_DBA3 sbo:Latency "0.0"^^xsd:double
<http://ethereum.ethstats.io/Block_0> owl:topDataProperty 0
<http://ethereum.ethstats.io/Block_1920000> ethon:BlockDataProperty 1920000
sbo:_DBA2 sbo:NbMbDevPerUser "1000.0"^^xsd:double
<http://ethereum.ethstats.io/Block_1920000> ethon:NetworkObjectProperty ethon:
    ProtocolVariant_DAO
sbo:_DBA1 sbo:Cost "2.0"^^xsd:double
<http://ethereum.ethstats.io/Block_1150000> ethon:BlockDataProperty 1150000
sbo:_DBA3 sbo:NbTxDevPerUser "0.0"^^xsd:double
sbo:_MaticMainnet sbo:Cost "5.0"^^xsd:double
sbo:_DBA2 sbo:Latency "2.0"^^xsd:double
sbo:_DBA1 owl:topDataProperty "10.0"^^xsd:double
 SameIndividual: ethon:ProtocolVariant_IceAge
```

Listing A.1 : Axiomes de l'ontologie présentée dans le chapitre 4