IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received March 28, 2022, accepted April 28, 2022, date of publication May 9, 2022, date of current version May 13, 2022.

Digital Object Identifier 10.1109/ACCESS.2022.3173313

A Blockchain Ontology for DApps Development

LÉO BESANÇON[1], CATARINA FERREIRA DA SILVA[2], PARISA GHODOUS[1], AND JEAN-PATRICK GELAS[1]

[1]LIRIS, Univ Lyon, Université Claude Bernard Lyon 1, 69100 Villeurbanne, France
[2]Instituto Universitário de Lisboa (ISCTE-IUL), ISTAR, 1649-026 Lisboa, Portugal

Corresponding author: Léo Besançon (leo.besancon.38@gmail.com)

The work of Léo Bensançon was supported by B2Expand, Lyon, France.

ABSTRACT Decentralized Applications, or DApps, provide distributed trusted applications that use blockchains. They are often composed of several services, such as transaction scalability protocols, decentralized storage and distributed computing solutions. In order to help formalize these applications, facilitate their development and improve their interoperability, we propose a novel blockchain Ontology focused on the concepts involving DApps. This ontology extends the existing EthOn ontology. It defines several key concepts related to DApps development, as well as the relations between these concepts. It features the formalization of known use cases and design patterns of blockchain technology through blockchain patterns. We use Semantic Web Rule language (SWRL) in order to define rules that express constraints on the formalized concepts. We then execute an inference engine and obtain new constraints on the properties of a defined DApp, such as its cost, based on the DApp characteristics and the services it uses. For illustration we show the inference of constraints between the Ethereum blockchain and its sidechain Polygon. We apply our research work in the field of blockchain video games. This application shows how to use the ontology to model DApps, and can be adapted to other fields.

INDEX TERMS Blockchain, interoperability, decentralized applications, DApps, ontology.

I. INTRODUCTION

Blockchains are a disruptive technology. They are the foundations for Decentralized Applications (DApps) with interesting properties: transaction transparency, application auditability and censorship resistance [1]. It has applications in various fields, such as finance [2], healthcare [3] or video games [4] and has evolved a lot since the creation of Bitcoin in 2008.

However, as seen in [5], the interoperability within blockchain systems is one of several challenges the technology faces today. Interoperability has an important role in maturing the blockchain industry. This industry is composed of various organisations, platforms, projects and services which can be improved greatly by interacting with other systems. We consider blockchain interoperability in three ways. The first type of blockchain interoperability is between different blockchain systems such as Bitcoin and Ethereum. Its goal is to transfer information and value between these blockchain systems. A second type of interoperability is between projects using the same blockchain system. As an example, projects on the Ethereum blockchain may need to use common standards and interfaces, such as Ethereum Request for Comments, to allow for a better integration between these projects. Finally, the interoperability between the services that compose a DApp is also needed [6]. A DApp may need to implement several components that have to interact with each-other, for example distributed storage or scalability solutions.

Our paper focuses on this last type of blockchain interoperability. We find that interoperability research for DApps development is lacking. As a result, we focus on the semantic interoperability of services used within a DApp.

Our contributions are as follows:

- We propose a DApps ontology, enabling the formalization of the concepts related to DApps and their relations. This includes the formalization of the concepts related to blockchain services, which support an operative DApp. This ontology extends an existing blockchain ontology, EthOn [7].
- We show the modeling capabilities of our DApps ontology by exemplifying and extending our formalisation with several blockchain design, which are

The associate editor coordinating the review of this manuscript and approving it for publication was Thanh Ngoc Dinh.

common design patterns in blockchain DApps such as Non-Fungible Tokens (NFT) management App, NFT marketplaces, Decentralized Autonomous Organization (DAO), blockchain oracles, supply chain and blockchain non-blocking user interfaces.
- We specify Semantic Web Rule Language (SWRL) rules to facilitate the coherence verification of our DApps ontology. They aim to constrain the concepts of our ontology. These rules are indeed used to infer new axioms for the ontology.
- We show the application of these DApp ontology and SWRL languages in helping the interoperability between two blockchain platforms.
- We exemplify and validate these contributions within the Light Trail Rush (LTR) industrial video game.

The remainder of the paper is organised as follows. Firstly, a literature review is presented on blockchain ontologies and their capabilities. We also analyze various blockchain services used within a DApp. Then, we propose a novel ontology that focuses on formalizing DApps. We additionally provide the modeling of blockchain patterns [8]. Then, we present Semantic Web Rule Language (SWRL) rules. We then present the applications of our research in the Video Game Industry, followed by the results and discussions. Finally, we present our conclusions, which include the perspectives of our work.

II. CURRENT STATE OF THE ART

A. SEMANTIC INTEROPERABILITY IN THE BLOCKCHAIN INDUSTRY

The semantic interoperability of blockchains is a complex subject as many blockchains seemed incompatible at a fundamental level until recently [5].

One example is how different blockchains propose different characteristics for what is called finality. Finality of a blockchain transaction occurs when it has been confirmed by the network and can no longer be modified. The Proof of Work (PoW) [9] consensus algorithm, used by many blockchains such as Bitcoin, is inherently probabilistic. Even when the network is not attacked by malicious actors, it is possible for two miners to create a valid block simultaneously. It is thus possible that a block considered valid at one point is deleted or modified. This is called probabilistic finality.

Other consensus algorithms, such as Proof of Authority (PoA) [10], have an absolute finality, which means that once a block is considered valid by a node of the network, it cannot be modified or deleted from the blockchain.

These two types of blockchains are then incompatible. Indeed, if we wish to transfer information from a blockchain with a probabilistic purpose to a blockchain with a certain purpose, it is possible to validate the information and transmit it even though it will be invalidated later.

However, there are probabilistic ways to link a blockchain without immediate finality with a blockchain with immediate finality, such as those presented in the publications [11] and [12]. From a semantic point of view, these allow to automatically translate incompatible concepts under specific hypothesis.

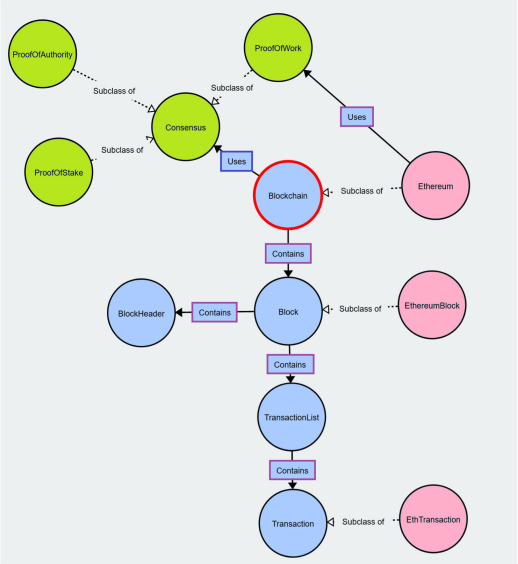


FIGURE 1. Example of a blockchain ontology.

Furthermore, some terminologies related to smart contracts, such as control flow graphs or states, are formalized by Chatterjee *et al.* [13]. This kind of work can be useful to interconnect two blockchains supporting smart contracts, but also to connect a blockchain with other systems interacting with smart contracts on this blockchain.

Finally, other works, such as blockchain Ontology with Dynamic Extensibility (BLONDiE) [14] and Ethereum Ontology (EthOn) [7], aim at developing ontologies on the blockchain ecosystem. The blockchain ecosystem is composed by all of the blockchain development projects. However, these projects are incomplete hindering consistent and non-trivial implementations of semantically compatible blockchains.

Fig. 1 presents our proposal of a simple blockchain ontology. Different concepts, and also their relations, are defined thanks to Web Ontology Language (OWL) [15]. Here, we have modeled a small part of the concepts and relations related to the Ethereum blockchain. This ontology models the following elements:

- Blockchain contains blocks,
- Block contains a block header and a transaction list,
- Transaction list contains transactions,
- Blockchain uses a consensus algorithm,
- Finally, Ethereum is a blockchain, which uses the PoW as a consensus algorithm.

Sandra [16] is an ontology engine created by the company EverdreamSoft. The company uses this engine to build the Crystal Spark Cannon library, integrating blockchain concepts such as addresses or collectable assets of a game, the Non-Fungible Tokens (NFTs). NFTs are representations on

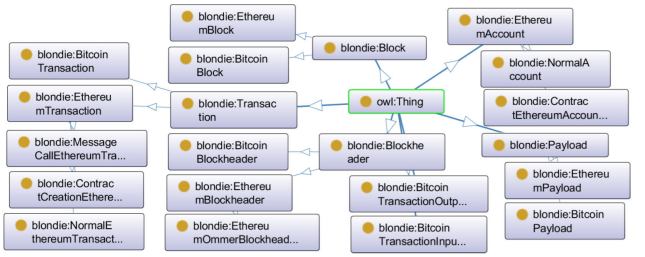


FIGURE 2. **Visual representation of the BLONDiE ontology.**

the blockchain of game assets. Players can buy, sell or trade them freely. The possession of an NFT is then reflected in the game.

The main difference between Crystal Spark Cannon and the two previous ontologies is its use within industrial projects and within the EverdreamSoft company developing the tool. Thus, it is possible to carry out simple queries such as retrieving all the assets associated with a given address. Based on the address format given, the tool then automatically deduces the blockchain on which to perform the query and provides as output all the elements found.

We have seen previously that ontologies are useful to describe blockchain systems. Indeed, ontologies aim at semantically defining the different concepts needed in a given domain. Blockchain systems often involve different domains that use similar concepts, which may not have consistent definitions. Therefore, several works try to semantically define what a blockchain is.

For example, BLONDiE [14], Fig. 2, and EthOn [7], Fig. 3 use OWL to describe such ontologies. They are useful for getting a global understanding of how different blockchain concepts such as transactions, address and signatures are related to each other, as well as for formalizing these concepts, but we did not find any applications using these ontologies. However, the PHP framework Sandra [16] allows users to easily design their blockchain ontologies, and is used by EverdreamSoft's Crystal Spark Cannon to query blockchain assets.

One downside of using Sandra is the lack of integration with existing tools for OWL ontologies. This is why our ontology focuses on extending EthOn instead.

Through the use of ontologies, the different systems that interact with and within a DApp all have the same definition of the concepts and data structures they use, which improves the semantic interoperability of a DApp.

a: BLONDIE

BLONDiE [14] seeks to formalize the basic concepts related to the Bitcoin and Ethereum blockchains. This ontology presents the blockchain as a data-only structure, detailing the composition of the blocks, and the transactions included in these blocks.

One of the interests of this ontology is to formalize the differences between these two blockchains. For example, Bitcoin is based on a transaction model by Unspent Transaction Output (UTXO) [17], so BLONDiE defines the concepts of BitcoinTransactionOutput and BitcoinTransactionInput. Since Ethereum is based on an account and account balance model, those two concepts have no meaning within the Ethereum ecosystem, and are therefore not specified for the Ethereum realm within BLONDiE.

On the other hand, currently, this ontology does not propose any formalization related to the protocols governing these two blockchains. This is an important limitation, since it restricts the possible applications of such an ontology.

b: ETHON

EthOn [7] only seeks to define concepts related to the Ethereum blockchain, but goes into more detail about the protocol. For example, the relationship of mining a block by an Ethereum account is defined within EthOn.

One of the limitations of this ontology is that it is restricted to concepts related to the Ethereum blockchain itself, and therefore does not formalize the whole ecosystem around the blockchain. For example, the various blockchain services such as scalability solutions of *layer two* [18] or cryptocurrency wallets [19] are not modeled within EthOn.

c: CONCEPTS RELATED TO BLOCKCHAIN SERVICES

Existing blockchain ontologies, such as BLONDiE and EthOn, do not specify concepts related to blockchain services. However, these concepts are fundamental to properly formalize a DApp. Sandra and Crystal Spark Cannon do provide this option, but they lack the expressive OWL environment. We thus propose in section III a blockchain ontology focused on DApps and associated blockchain services.

B. BLOCKCHAIN SERVICES

Blockchain services are projects that can be integrated by other blockchain projects in order to change the characteristics of the application. For example, a blockchain service can provide the scalability of the number of transactions that can be handled by the application, or a storage solution for the data of the application.

1) SCALABILITY

There are many different approaches to increase the capabilities of a given blockchain. These solutions are called *Layer 2 scalability*, since they are overlays of a blockchain. Thus, for an application running on Ethereum, it is possible to integrate technologies that improve the characteristics of the application. We can distinguish the proposed approaches as follows:

- Solutions that check the validity of transactions sent through *Validity Proofs*,
- Solutions requiring proofs of invalid transactions, called *Fraud Proofs*, which must be provided within a specified time period. This deadline is called the challenge period, as mentioned by Warren and Bandeali [20].

In addition, we also differentiate between:

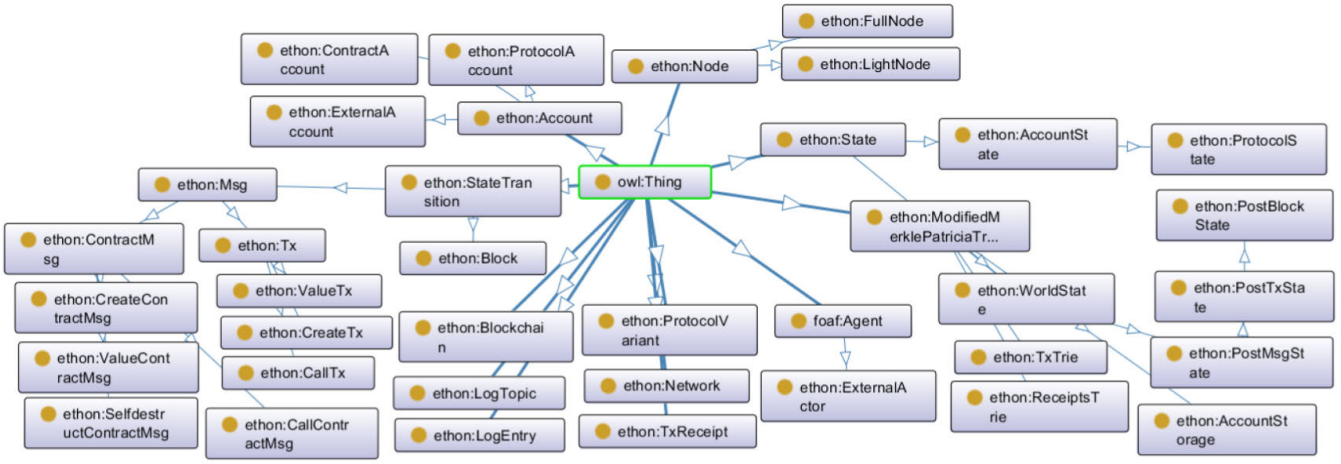


FIGURE 3. Visual representation of the EthOn ontology.

- Solutions with off-chain data storage such as state-channels [21],
- Solutions with on-chain data storage such as rollups [22],

Finally, Buildblockchain Tech and Avihu Levy [23] propose to classify the existing Layer 2 features according to these two characteristics, as shown in the table 1.

TABLE 1. Table of characteristics of different Layer 2 scalability solutions, adapted from [23].

Verification / Data storage	Validity Proofs	Fraud Proofs
On-chain	ZK-rollups	Optimistic rollups
Off-chain	Validium	Plasma

a: STATE-CHANNELS

State-channels consist in the exchange of off-chain messages between the parties involved. In case of conflict between these parties, each one can publish on the blockchain the transmitted messages, and a smart contract will verify the exchanged information. The concept of state-channel is for example used by FunFair [24], but only between two participants. A schematic version of this concept is presented Fig. 4.

More general approaches exist, such as the Lightning Network [21] for the Bitcoin blockchain, the Raiden Network for Ethereum, and Counterfactual [25]. However, these approaches require additional research. For example, for the Lightning Network and the Raiden Network, research is needed to find optimal routing of transactions across nodes that form users on the network. Fig. 5 explains how these networks can connect two nodes through a channel in an indirect way.

One of the characteristics of state-channels is the need for cash to run the system. Thus, it is likely that *hubs* will be created to offer this liquidity for a fee.

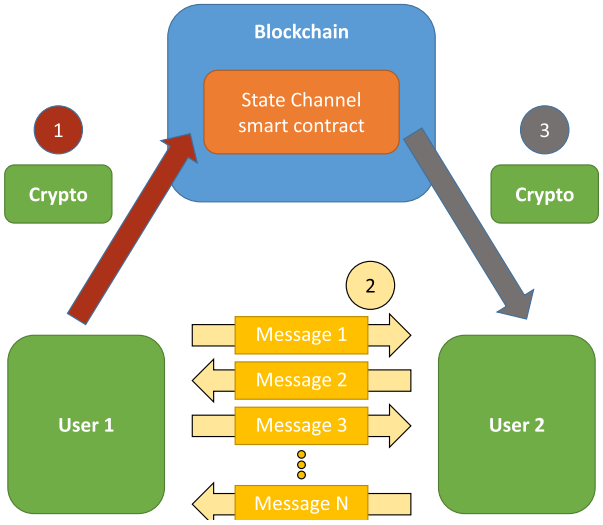


FIGURE 4. Explanatory diagram of the functioning of state-channels.

b: ROLLUPS

Rollups [22] are a type of scalability solution aiming at compressing blockchain transactions. To do this, different techniques can be used, such as aggregating transactions into *batches*. Another technique is to store a certain amount of data within off-chain transactions. However, some data for each transaction remains on the blockchain.

The concept of Rollups is depicted in Fig. 6. Transaction aggregation allows to compress data efficiently. For example, a multi-signature, which is a single signature that is valid if all the signatures of the underlying transactions are valid, is smaller than all the individual signatures.

In addition, as we will see, there are two main methods for increasing the scalability of transactions through rollups: Zero-Knowledge Rollups (ZK-rollups) and Optimistic rollups.

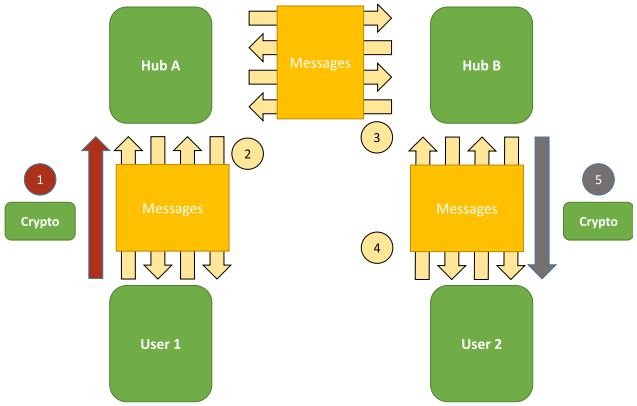


FIGURE 5. Explanatory diagram of the functioning of the Lightning Network and the Raiden Network.

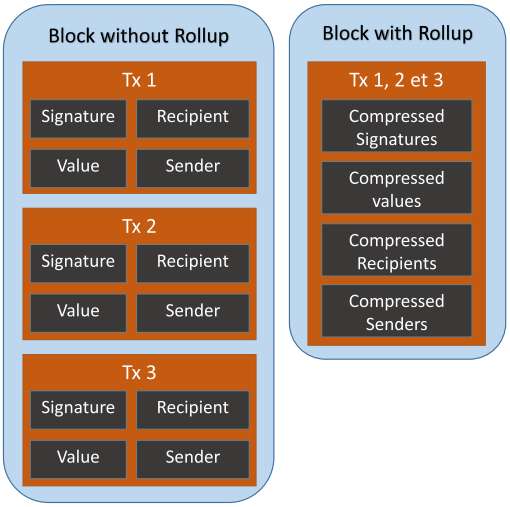


FIGURE 6. Explanatory diagram of how Rollups work.

c: ZK-ROLLUPS

ZK-rollups are based on the use of Zero Knowledge Proofs (ZKP). ZKPs allow to prove information about data without disclosing other information.

ZK-rolls use this concept to compress transaction data, while proving the validity of the compression. It is therefore the use of *Validity Proofs*.

Loopring [26] is an example of a project using ZK-rolls to increase the number of transactions possible on their decentralized exchange.

An example of a project that enables scalability of DApps is ZKSync [27], a Layer Two platform for Ethereum that also uses ZK-rollups.

d: OPTIMISTIC ROLLUPS

Unlike ZK-rollups, Optimistic rollups do not rely on formal proof of validity of compressed transaction data. Instead, so-called *Fraud Proofs* are used. This means that if no evidence of the invalidity of a transaction is provided within a specified time, then that transaction is declared valid.

An example of an optimistic rollup project is Arbitrum [28].

e: SUMMARY ON ROLLUPS

Buterin [22] quantitatively compares the characteristics of these two types of rollups.

In addition to the differences between the two types of rollups presented here, each rollup must choose certain characteristics that may impact the properties of the rollup. For example, the centralization, or not, of the transaction submission process. The centralization of this process makes it possible to simplify it, but also makes it possible to censor certain transactions, since a single actor chooses which transactions are included in the *batch*.

f: VALIDIUM (StarkEx)

StarkEx [29], developed by StarkWare, is a ZKP-based scalability solution. It can be used as ZK-rolls, keeping the data on chain.

However, StarkEx can also be used off-chain. The data availability problem is circumvented via a *Data Availability Committee*, which must ensure the availability of any data included in the Merkle Tree of a checkpoint.

We had mentioned Loopring, a decentralized marketplace using ZK-rollups. One of their competitors, DeversiFi [30], is based on StarkEx, with the use of Validium instead of ZK-rollups.

g: SIDECHAINS

A sidechain is a blockchain, which is linked to another blockchain, called mainchain by a *two-way peg* [31]. This blockchain is usually used for a single application, such as a video game. It is then possible to use a consensus method different from the parent blockchain, such as PoA [10], DPoS [32], or PoS [33]. The advantage of using a sidechain and not a separate blockchain is that if the daughter blockchain consensus fails, users can prove on the parent blockchain what happened on the daughter blockchain. A generic sidechain design, called Plasma [34], is implemented in several projects. We will focus on two existing sidechain solutions, Loom Network [35] and Polygon [36].

Finally, sidechains are one of the most flexible scalability solutions for developers. Fig. 7 shows the basic principle of sidechains. This principle has four main steps:

1) A user transfers a cryptocurrency A to the sidechain deposit contract on the main chain.
2) The sidechain will then automatically allocate the deposited funds to the user.
3) The user can then use these funds on the sidechain, at a greatly reduced cost compared to transactions on the main blockchain. Regularly, a *checkpoint* of the last blocks of the sidechain is created and submitted to the main blockchain. This checkpoint is obtained by hashing these blocks, and contributes to the security of the system.

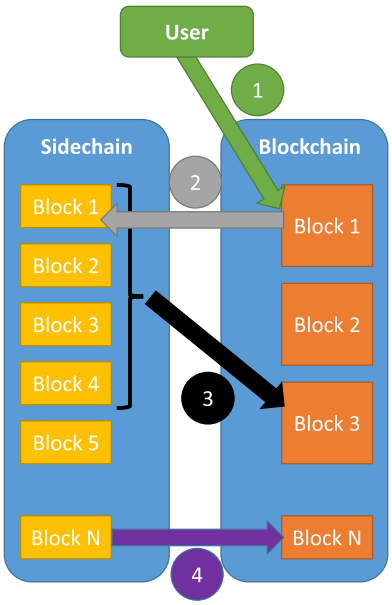


FIGURE 7. Explanatory diagram of how sidechains work.

4) Finally, the user can retrieve his funds from the main chain at any time. A certain delay is however necessary before the transfer is effective in order to leave time for challenges in case of disputes. Indeed, since sidechains work by *Fraud Proofs*, if a user tries to validate invalid transactions, another user can bring proof of the invalidity of these transactions. A certain amount of time must be allowed to make this principle work.

h: POLYGON

Polygon [37], rebranded from Matic [38] is a sidechain linked to Ethereum, working by *Fraud Proofs*, and allowing for example the *mint* of tokens on Polygon before sending them back to Ethereum. The *mint* is the generation of a new copy of a token. We detail the features of Polygon later, in section V-A, as we have studied its operation in detail to validate our contributions within a game.

i: LOOM

Loom [35] started as another Ethereum-related sidechain, dedicated mainly to the video game world. Today, they are developing a multi-chain platform.

j: MULTI-CHAIN

Multichains are scalability solutions using interoperability between different blockchains through Bridges [39]. As shown in Fig. 8, we can sort of see these solutions as a generalization of sidechains. Each of the four blockchains shown on this diagram is indeed linked to another one, called *Hub* or *Relay Chain* by the same bidirectional anchoring mechanism. The "daughter" blockchains are called *shards*.

Thus, instead of having a parent blockchain linked to a daughter blockchain, a set of blockchains can be linked

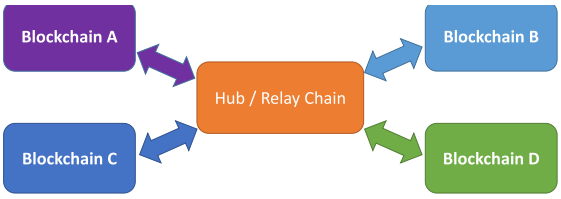


FIGURE 8. Schematic representation of the operation of multichains.

together by a common validation layer. However, the principles of communication between the *shards* of a multi-chain vary between different implementations of this principle, which we will now detail.

k: POLKADOT

Polkadot [39] proposes a multichain running on their consensus algorithm called Nominated Proof of Stake (NPoS). This algorithm is a hybrid between PoS and DPoS.

A particularity of Polkadot is that each *shard* has the same security. The security of the multi-string is therefore cooperative, which means that each *shard* has the same validators.

Polkadot proposes the use of a main chain, called Relay Chain, to facilitate communication between the other chains.

l: COSMOS/TENDERMINT

Cosmos [40] is a project very similar to Polkadot, and works by PoS. The main difference is on the validation model of the different chains. Unlike Polkadot, Cosmos offers competitive security. This means that each chain manages its own validation, and must therefore take into account the security of the other chains with which it communicates.

Just as Polkadot proposes a relay chain, Cosmos proposes the use of particular chains called hubs. However, several different *hubs* can be used, and their use is intrinsically necessary. Indeed, it is these *hubs* that implement the necessary bridges between the different *shards* of Cosmos.

m: HARMONY

Harmony [41] proposes a multi-chain by sharding, via a validation in PoS. Their bridge with Ethereum allows, concretely, to use Harmony as a multitude of sidechains linked to Ethereum.

2) DATA STORAGE

a: InterPlanetary FILE SYSTEM

InterPlanetary File System (IPFS) [42] is a free, peer-to-peer distributed storage system. It works in a simple way: anyone can choose to create an IPFS node on their system, and add files to it.

On IPFS, each file is represented by its hash, so that anyone can access a resource stored on IPFS for free through its hash. Thus, to access a resource, an IPFS node will propagate the request to neighboring nodes. The set of nodes being a mesh of the network, after a certain routing, the request is propagated to a node which has a resource with the same hash

as the request. This node will then transmit the data to the requesting node.

Fig. 9 shows how an IPFS user can obtain a resource that they do not yet own, based on its simple hash. In this example, user 1 is looking for a file whose hash is *Hash*1, which only user 3 owns. However, users 1 and 3 are not directly connected, they must communicate with the user 2, intermediary of the requests. Steps 1 and 2 correspond to network discovery: the nodes of the network propagate the file request requests until they find a user who owns the right file. Once a client with the desired file is found, steps 3 and 4 correspond to sending the file, by traversing the file request path in reverse.

This system has interesting properties, since it is free. Moreover, the risk of losing access to a data is mitigated, since an actor wishing to have a data accessible on IPFS can very well host an IPFS node himself. However, other resources may become inaccessible, since no financial compensation is offered to those who share their storage capacity with the rest of the network. In addition, the latency of this peer-to-peer system is quite high, averaging 7 seconds [43].

b: FILECOIN

Filecoin [44] is an overlay to IPFS that offers financial compensation to participants. This solution can be seen as sales made on the blockchain of distributed storage capacity.

c: STORJ

Storj [45] is another distributed storage solution. Unlike IPFS or Filecoin, it is more akin to a decentralized cloud, since data is replicated, clustered, and encrypted.

C. SYNTHESIS

Many services, of different types, offer solutions to improve the DApps ecosystem. Some of these services can change the cost and latency characteristics of a blockchain. This is the case of sidechains, state-channels, rollups or multichain. These different technical solutions have their advantages and disadvantages. For example, state-channels are easier to implement than multichains, but they can only be used for two-party interactions. Moreover, the different instances of these services make different implementation choices. This is for example the case of Loopring and Arbitrum. Both are instances of rollups, but Loopring relies on ZKPs to validate changes in the state of the blockchain, while Arbitrum relies on Fraud Proofs, which means that in case of conflicts one of the actors must propose a formal proof of error. Other services also offer new functionalities that would not be possible via a simple blockchain, such as distributed computing or distributed data storage.

Each of the presented solutions has different properties and constraints, in terms of cost, security, decentralization, maturity or even possible application cases. This means that choosing which services to use to build a DApp is complex, and requires to study all the suitable services. Thus, to fully understand these properties and constraints, we formalize how they work and their use cases, in the next section. Table 2 presents some of the trade-offs involved in the choice of a scalability solution for a given DApp.

III. PROPOSED ONTOLOGY

We have described in the previous chapter the interest of having a formalization of the different concepts of a DApp within an ontology. In order to facilitate the construction of an ontology specific to an application, we propose a generic blockchain ontology. This ontology can then be extended when we want to formalize a DApp. In this ontology, we use the Decentralized blockchain Applications (DBA) terminology to formalize DApps in order to avoid ambiguities between these applications and decentralized applications that do not integrate the blockchain technology.

A. GLOBAL VIEW OF THE ONTOLOGY

With the help of the OntoGraph Plugin for Protégé, a partial graph of the ontology is in Fig. 10. In this graph, we can see one of the fundamental concepts related to blockchains: transactions. A blockchain transaction is a message signed by a user, which will be added to a block of the blockchain. A transaction can be *pending*, i.e. waiting for validation, or *settled*, i.e. validated or invalidated. Once validated, a transaction cannot be removed from the blockchain. Another concept shown in this figure is the blockchain, which Polygon and Ethereum inherit. For the sake of readability, we do not detail in this figure all the other concepts of the ontology, such as DApps, transaction scalability solutions or consensus methods. However, figures detailing restricted parts of the ontology are presented later.

We have previously presented two existing blockchain ontologies, BLONDiE and EthOn. In order to build our ontology, we study the possibility of extending these existing ontologies.

The EthOn ontology formalizes many concepts related to the Ethereum blockchain. For example, it formalizes the concepts of Ethereum transactions and Ethereum blocks, but also contract calls and their creation, as well as the changes of state of the blockchain, and more generally the data structures used by Ethereum and their uses. We have therefore chosen to extend the EthOn ontology, since it already has many important concepts for the formalization of DApps.

BLONDiE is the other existing blockchain ontology. It is interesting since it formalizes different blockchains: Ethereum, Bitcoin, and the private blockchain Hyperledger Fabric. However, the concepts presented are not detailed enough. The transactions of each blockchain are defined, but little information about their use within the blockchain is present. For example, the calls or creations of Ethereum smart contracts are formalized as it is the case within EthOn, but not the data structures used for each blockchain. Thus, the concept of *Merkle-Patricia Trie*, which defines the tree storing the set of transactions and the associated state changes, is present in EthOn but not in BLONDiE. For this reason, we do not extend the BLONDiE ontology. On the other

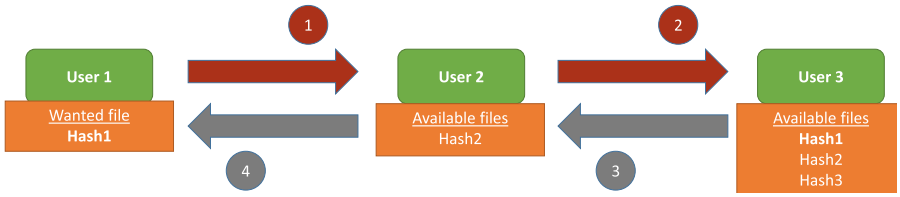


FIGURE 9. Explanatory diagram of the functioning of IPFS.

TABLE 2. Table summarizing some advantages and disadvantages of the main methods of scalability of blockchain transactions.

Scalability solution	Advantages	Trade-offs
State-channels	Ease of implementation Negligible costs	Collateral needed Complex for $n \geq 3$ parties
ZK-rollups	Same security as the underlying blockchain	Complexity of Zero-Knowledge Proofs
Optimistic Rollups	Ease of implementation	Challenge period needed
Sidechains	Ease of implementation Many projects have functioning sidechains	Challenge period needed
Multichains	Allows for interoperability between numerous blockchains	Complexity of the system

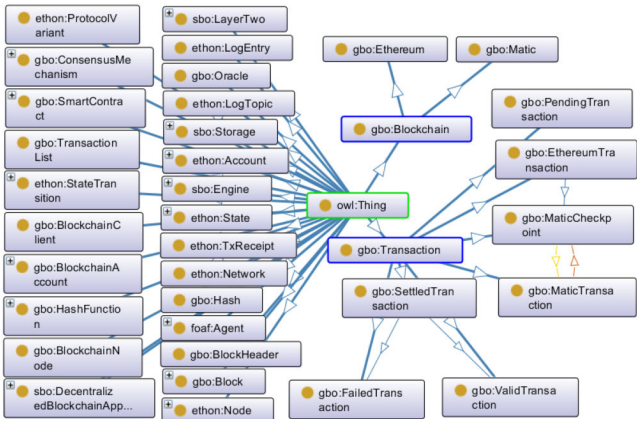


FIGURE 10. Partial visualization of our ontology. Only the transaction and blockchain concepts are detailed for the sake of readability.

hand, we still used this ontology to understand how concepts from different blockchains can coexist within the same ontology.

The proposed ontology uses the following prefixes:

- ethon, for the imported EthOn ontology.
- gbo, to formalize the generic concepts of the blockchain. With the help of the OntoGraph Plugin for Protégé, a graph showing the generic part of the ontology is presented Fig. 11.
- sbo, to formalize the concepts specific to DApps. A graph showing the DApp-specific part of the ontology is presented Fig. 12.
- bpo, to formalize different blockchain patterns. The part dedicated to the blockchain patterns of the ontology is presented Fig. 13. We will then list all the blockchain patterns that we have formalized, as well as examples of their application.

B. CLASSES

This subsection details the set of main classes in our ontology. Each class represents a concept related to blockchain or DApps, and can be described by an annotation, constraints, as well as with example individuals of that class. The constraints of each concept can be linked to another concept of the ontology, or not. In the first case, the ontology will represent the constraint through object properties or *ObjectProperties*. Otherwise, the ontology will represent it through a data property or *DataProperties*.

Hereafter we present some of the concepts, including their annotation in natural language, we formalize in the our blockchain ontology extension.

1) GBO:BLOCKCHAIN

a: ANNOTATION

According to NIST, a blockchain is a collaborative ledger that is resistant to tampering and maintains transactional data [46].

b: CONSTRAINTS

The constraints of a gbo:Blockchain, described in the following list, represent all the data and classes related to this concept. Concretely, these constraints will manifest in our ontology as DataProperties and ObjectProperties.

- Uses a Consensus Method, described in the class gbo:ConsensusMechanism
- Contains Blocks, described in the class gbo:Block
- Average time to create a block (seconds), the average time between each block
- Block size (bytes), the maximum size of each block
- Possibility of smart contracts (0/1)
- Finality (seconds, estimated confirmation time for a fixed certainty threshold)
- Complexity of operations (if smart contracts possible)
- Cost of an operation (€/op)

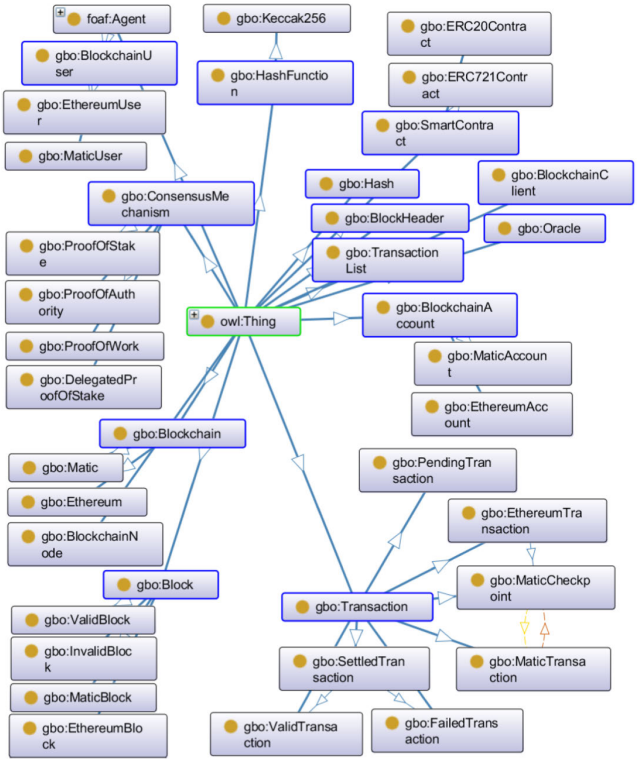


FIGURE 11. Visualization of the generic part of our ontology, including the main concepts related to blockchains.

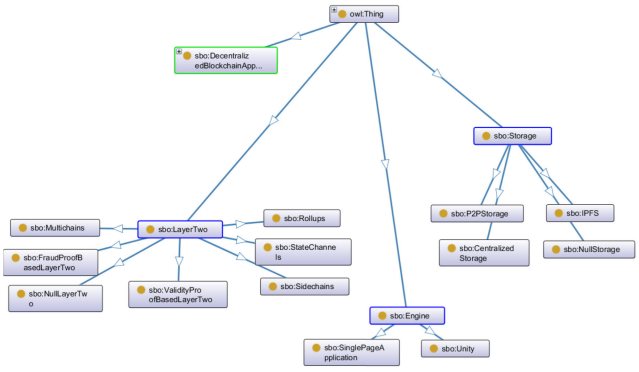


FIGURE 12. Visualization of the DApp-specific part of our ontology.

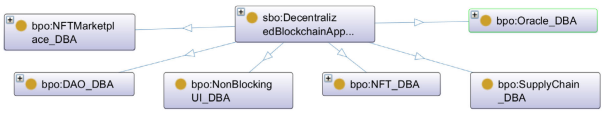


FIGURE 13. Visualization of the part dedicated to the blockchain patterns of our ontology.

The constraints listed here correspond either to the blockchain as a data structure (e.g. block size) or to the characteristics of the protocol (e.g. the finality).

c: EXAMPLES OF POSSIBLE INDIVIDUALS

- 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€

2) GBO:BLOCK

a: ANNOTATION

A block is the atomic unit of state changes within the blockchain. It is cryptographically linked to the next block by its hash.

b: CONSTRAINTS

- Contains a list of transactions, whose concept is formalized in the class gbo:Transaction
- Size of a Block (bytes)

c: EXAMPLES OF POSSIBLE INDIVIDUALS

- The first block of a blockchain is special because it cannot reference previous blocks. It is called the *Genesis block*.

3) GBO:TRANSACTION

a: ANNOTATION

Validated transactions are contained in a block. Before being included in a block, a transaction is said to be ''awaiting validation,'' and belongs to a *mempool*.

b: EXAMPLES OF POSSIBLE INDIVIDUALS

- For each block, a particular transaction is called the *coinbase transaction*. This is a transaction that pays the miner of this block. This type of transaction is the only one that allows a monetary creation.
- The sidechain Polygon needs to realize regular *checkpoints* on Ethereum [47]. As described in paragraph II-B1.g II-B1g Sidechains., these checkpoints are transactions on Ethereum, and allow the Polygon transactions included in the checkpoint to get the same security as if they were regular transactions on Ethereum. Ethereum uses a more secure consensus algorithm than Polygon, so these checkpoints are fundamental to the proper functioning of the sidechain.

4) GBO:BLOCKCHAINUSER

a: ANNOTATION

A user of a blockchain is a human who owns and manages an account on this blockchain. The access to this account can be done through a third-party service such as Metamask, or directly by the user if they own a node of the blockchain. A user can be a cryptocurrency owner, a DApp user, or a miner. A miner, for a blockchain operating by PoW, participates in the creation of new blocks, secures the blockchain, and is paid for this task.

5) GBO:CONSENSUSMECHANISM

a: ANNOTATION

A consensus mechanism is an algorithm for solving the Byzantine Generals Problem [48]. In particular, it is used to

define which blocks and transactions will be considered valid by the network.

b: EXAMPLES OF POSSIBLE INDIVIDUALS

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

6) GBO:SMARTCONTRACT

a: ANNOTATION

In our ontology, a smart contract is used by a DApp and is included in a blockchain.

In addition to decentralized transfers of monetary value through crypto-currencies, blockchains can also act as information registers, being able to provide, for example, identity management and authentication of a person. Another feature of some blockchains is the ability to run programs on the blockchain. These programs are called "smart contracts," and their execution has the same properties as other transactions: immutability, pseudonymity, traceability and reliability. The smart contract concept, initially proposed in 1997 by Szabo [49], can benefit greatly from the properties of blockchains.

Vitalik Buterin proposed Ethereum [50] in 2014, the first blockchain dedicated to smart contracts. Hyperledger Fabric [51] is another example of such a blockchain, pushed by International Business Machines Corporation (IBM) and the Hyperledger consortium of the Linux foundation. The latter is a private blockchain, which means that the consensus algorithm used relies on the trust that users have in the network administrators. Thus, the execution of a smart contract on Hyperledger Fabric is not decentralized, unlike a smart contract on Ethereum. Other technologies belonging to the category of distributed ledgers, or Decentralized Ledger Technologies (DLTs), of which blockchains are a part, can offer smart contracts. El Ioini and Pahl [52] present the different types of DLTs that exist today, such as blockchains, *Tangle*, *Hashgraph* or *Sidechain*.

7) SBO:DECENTRALIZEDBLOCKCHAINAPPLICATION

a: ANNOTATION

A DApp is an application that uses blockchain technology. However, most existing DApps need other associated services to work, such as a *engine* that can act as a *front-end* for the application, storage solutions, or scalability solutions for the blockchain used.

In section V-A, we define several SWRL rules that allow us to infer the properties of a DApp from its characteristics and the technologies it uses.

b: CONSTRAINTS

- Uses a gbo:Blockchain
- Uses at least one gbo:SmartContract
- Uses an engine as defined in the following via the sbo:Engine class
- Uses a sbo:LayerTwo as defined hereafter
- Uses a sbo:Storage as defined hereafter
- Number of transactions to execute
- Running costs
- Operating latency
- Amount of data storage required for operation

We detail all the DataProperties related to this class in the section III-E.

8) SBO:ENGINE

a: ANNOTATION

Engines are a generic concept related to DApps. Depending on the DApp considered, the associated engine can be a web page, a game engine, an application management framework, etc. We define sbo:Engine as tools allowing to develop interfaces for a DApp.

9) SBO:LAYERTWO

a: EXAMPLES OF POSSIBLE INDIVIDUALS

- Polygon
- Zero-Knowledge Rollups
- Validium

10) SBO:STORAGE

a: EXAMPLES OF POSSIBLE INDIVIDUALS

- Centralized Storage
- IPFS
- Filecoin

11) GBO:POLYGONCHECKPOINT

a: ANNOTATION

A Polygon checkpoint is an Ethereum transaction that validates Polygon transactions. A new checkpoint is submitted every 30 minutes to synchronize Polygon with Ethereum.

b: CONSTRAINTS

- Contains a list of Polygon transactions validated by this checkpoint, referenced by their hash.

Appendix present additional classes of our ontology.

C. OBJECT PROPERTIES

ObjectProperties are relations between two of the concepts defined in the ontology. Here, we define several properties, related to the blockchain as a data structure (e.g. gbo:Contains or gbo:IncludedIn), or to functional usage of actors or services (e.g. gbo:Uses).

1) GBO:CONTAINS

The relationship gbo:Contains relates two objects when one contains the other, from the point of view of data structures. Thus, a blockchain contains blocks, and blocks contain transactions.

2) GBO:INCLUDEDIN

The relationship gbo:IncludedIn is the inverse of the relationship gbo:Contains. Thus, a valid block is included in a blockchain.

3) GBO:USES

The gbo:Uses relation is rather generic. It can represent an actor who uses a system. For example, a blockchain user will use a blockchain.

This relation is also suitable to indicate that a technology needs another one to work. For example, a blockchain uses a consensus method, and a DApp uses a blockchain.

4) GBO:NEXTBLOCK

The gbo:NextBlock relation is used to reference the next block that is mined by a blockchain at a certain time.

5) GBO:LASTCHECKPOINT

The gbo:LastCheckpoint relation allows a blockchain that uses a checkpoint system on another to reference the last checkpoint created. For example, the Ethereum Polygon sidechain regularly publishes checkpoints on the Ethereum blockchain.

6) GBO:MINESFOR

The gbo:MinesFor relation connects a blockchain user to a blockchain where he is a miner. This relation can only involve blockchains with miners, thus using PoW as a consensus method.

7) GBO:HASBENEFICIARY

The relation gbo:HasBeneficiary links a valid block of a blockchain to the miner who mined it.

D. INDIVIDUALS BY CLASS

We define the individuals we need to properly formalize a complete DApp. In particular, these individuals are necessary to apply the SWRL rules that we define in section V-A.

1) GBO:BLOCKCHAIN

a: SBO:_POLYGONMAINNET

This individual represents the main Polygon blockchain.

b: SBO:_ETHEREUMMAINNET

This individual represents the main Ethereum blockchain, in its current version (ETH 1.0).

c: SBO:_ETHEREUMROPSTENTESTNET

This individual represents a test version of the Ethereum blockchain. This version, called Ropsten, runs by PoW, like the main chain, and thus replicates the behavior of the main chain quite well. As with the other testnets, the main difference with the Ethereum mainnet is that Ropsten's cryptocurrency (the Ropsten ETH) has no financial value. Thus, it is not possible to test or simulate the economic characteristics of a DApp with a testnet. Indeed, since cryptocurrency has no financial value in the testnet, a DApp actor who would have an interest in acting in a certain way on the mainnet may not behave in the same way on the testnet.

2) GBO:LAYERTWO

a: SBO:_POLYGONLAYERTWO

This individual is very similar to sbo:_PolygonMainnet. The difference is that it is considered here as a scalability solution for Ethereum, and not a simple blockchain. The associated properties will therefore be different.

b: SBO:_NULLLAYERTWO

This individual allows to consider a DApp which does not have a Layer Two. It will be useful when writing SWRL rules, in section V-A.

3) GBO:DECENTRALIZEDBLOCKCHAINAPPLICATIONS

We have detailed three different cases of DApps useful for B2Expand. The following three individuals represent DApps with the same objective, that of managing LTR assets. However, they use different technologies. The objective is to compare the characteristics of these individuals according to the technologies used.

a: SBO:_DBA1

This individual represents a DApp using only Ethereum, with no scalability solution in Layer Two.

b: SBO:_DBA2

This individual represents a DApp using Ethereum, as well as its sidechain Polygon.

c: SBO:_DBA3

This individual represents a DApp using only Polygon. Here, Polygon is therefore used as the main blockchain and not as a sidechain of Ethereum.

4) GBO:POLYGONTRANSACTION

We have defined three individuals that represent Polygon transactions submitted at different times.

a: SBO:_POLYGONTRANSACTION1

This transaction is submitted to the Polygon blockchain on 10/18/2021 at 6:20pm.

b: SBO:_POLYGONTRANSACTION2

This transaction is submitted to the Polygon blockchain on 10/18/2021 at 18:45.

c: SBO:_POLYGONTRANSACTION3

This transaction is submitted to the Polygon blockchain on 10/18/2021 at 18:50.

5) GBO:POLYGONCHECKPOINT

We have also defined two individuals that represent two successive Polygon checkpoints.

a: SBO:_POLYGONCHECKPOINT1

This transaction is submitted to the Ethereum blockchain on 10/18/2021 at 6:30pm.

b: SBO:_POLYGONCHECKPOINT2

This transaction is submitted to the Ethereum blockchain on 10/18/2021 at 7:00pm.

E. DATA PROPERTIES

DataProperties, are properties that link a class with basic data. This data can be integers, character strings, dates, etc.

We have chosen to formalize the main data properties for a few concepts, as well as those we need to apply SWRL rules in the V-A section.

DApp-specific DataProperties are divided into four categories: costs, transaction counts, data quantities, and latency.

First of all, the *Cost* data allows us to formalize the different costs of an application or of the services that make it up. We have chosen to differentiate the costs paid by the user from the costs paid by the developer of the DApp. Indeed, some developers may prefer to perform some transactions instead of the DApp users. This improves the user experience, as users do not have to pay transaction fees for certain features of the app. For example, many cryptocurrency exchanges pay the transaction fee of withdrawing a cryptocurrency to a user's wallet. The costs of storing data and transactions on a blockchain are also considered.

- Cost: Properties related to costs
  - CostDevPerUser: Developer Costs per user
  - CostUser: Cost for each user
  - CostPerMb: Cost for a megabyte of storage
  - CostTx: Cost of a transaction
    - CostBasicTx: Cost of a transfer of cryptocurrencies
    - CostTxTokens: Cost of a transfer of ERC-20 tokens, smart contract calls being more expensive than basic transfer of cryptocurrencies
    - CostTxDeploy: Cost of the deployment of a smart contract.

The *NbTx* data formalizes, for each DApp, an estimate of the number of blockchain transactions to be realized for a normal functioning of the application. This is an estimate since it is difficult to predict precisely the use that different users will have of the application. As with the differentiation of developer and user costs, we differentiate between transactions made by users and those made by the application's developers. As stated earlier, in some cases, a feature of a DApp will only be called by developers. Other features, such as the creation and purchase of ERC-721 assets on Ethereum, will require multiple transactions, performed by both the developer and users.

One of the objectives of this formalization being to formalize the costs of an application, we also differentiate the transactions that must be performed on a blockchain, from those that can be performed on a scalability service such as Layer Two. Indeed, the costs of transactions associated with these services can be very different. It is therefore important to know the constraints on the transactions in order to model all the costs as well as possible.

- NbTx: Properties related to the number of transactions needed for the application to function
  - NbTxDevPerUser: Number of transactions the developer needs to execute per user
    - NbTxDevPerUserOnMainchain: Number of transactions the developer needs to execute per user on the main blockchain
    - NbTxDevPerUserOnLayerTwo: Number of transactions the developer needs to execute per user on the Layer Two platform.
  - NbTxPerUser: Number of transactions each user needs to execute
    - NbTxPerUserOnMainchain: Number of transactions each user needs to execute on the main blockchain
    - NbTxPerUserOnLayerTwo: Number of transactions each user needs to execute on the Layer Two platform.

In a similar way to the number of transactions, the *NbMb* properties are intended to model the quantities of data necessary for the functioning of the application. As before, we differentiate between data stored by the developer and by the users. Indeed, in most applications, some data, such as user accounts, are stored by the developers, whereas other data, such as an executable of the application itself, are stored locally on the users' machines. We also differentiate between data on the blockchain and data that can be stored outside the blockchain.

- NbMb: Data properties related to the amount of data needed to be stored for the application to function
  - NbMbDevPerUser: Amount of data the developer needs to store for each user
    - NbMbDevPerUserOnMainchain: Amount of data the developer needs to store for each user on the main blockchain
    - NbMbDevPerUserOnLayerTwo: Amount of data the developer needs to store for each user on the Layer Two platform
    - NbMbDevPerUserOnStorage: Amount of data the developer needs to store for each user on the storage service.
  - NbMbPerUser: Amount of data each user needs to store
    - NbMbPerUserOnMainchain: Amount of data each user needs to store on the main blockchain
    - NbMbPerUserOnLayerTwo: Amount of data each user needs to store on the Layer Two platform

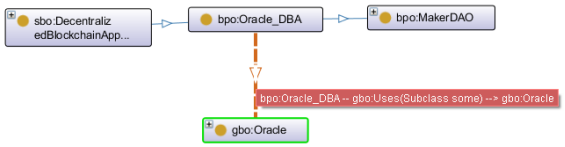


FIGURE 14. Modeling of an example DApp.

- NbMbPerUserOnStorage: Amount of data each user needs to store on the storage service.

Finally, the last type of DataProperties corresponds to the notions of latency of an application. As reads on a blockchain do not require the application of the consensus algorithm, it is important to differentiate the latency in reading or writing of the DApp and the services that compose it.

- Latency: Properties related to the latency of the application
  - LatencyReads: Read latency
    - LatencyReadsStorage: Read latency of the storage solution used by the application
    - LatencyReadsTx: Read latency of a transaction.
  - LatencyWrites: Write latency
    - LatencyWritesStorage: Write latency of the storage solution used by the application
    - LatencyWritesTx: Write latency of a transaction.

Figure 14 shows the example of the model of an application that uses an Oracle. We model the relation between the application and the oracle with an Object Property. In the following section, we present how to model different use cases such as this one, through design patterns.

IV. MODELING OF BLOCKCHAIN DESIGN PATTERNS

In order to show the possibility of modeling, thanks to our ontology and various use cases, we decided to model different design patterns existing in the blockchain industry. These patterns correspond either to types of blockchain use cases, or to design methods for a DApp element. The term blockchain pattern is taken up by Xu *et al.* [53] to study the *design patterns* used in the blockchain industry.

A. NFT MANAGEMENT DAPP

Many of the existing DApps aim to manage NFTs, usually symbolized by ERC-721 tokens. Most NFT management DApps, such as the one developed by B2Expand, create a new NFT by modifying the state of an existing contract.

However, some projects, such as Mintable [54], offer NFT creators two other alternatives. First of all, they have deployed a smart contract allowing the free creation of NFT, called *Gasless minting*. The created NFT is indeed on the blockchain, but the cost of changing the state of the contract is paid by the buyer of an NFT, only at the time of its purchase. By this system, a seller does not need to pay for the creation of unsold assets.

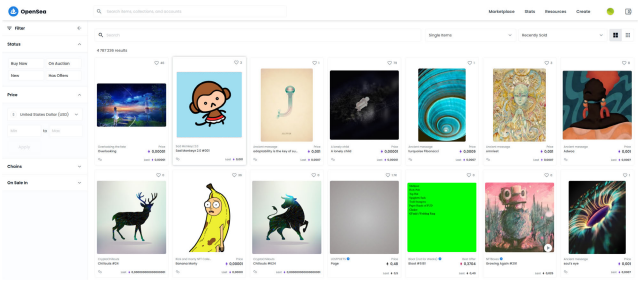


FIGURE 15. Example of an NFT Marketplace, https://opensea.io.

In addition, they offer another contract that makes it easy to deploy a sales contract, even for a single NFT. This also allows for more customization of each NFT. However, it also increases the costs considerably.

An example of a DApp that follows this blockchain pattern is CryptoKitties [55]. This DApp gives the possibility to players to collect virtual cats. These cats are NFT, and players can buy, sell, or breed them. Each CryptoKitty has a set of attributes, giving it a higher or lower financial value. Fig. 15 shows various NFTs on sale. Each on of these NFT can be used in a specific DApp built around them.

The DataProperties specific to NFT management DApps are as follows:

- DeployContractForEachNFT: Defines whether to deploy a smart contract for each NFT or not
- GaslessMinting: Defines whether the creation of NFTs is free or not.

These two properties are sufficient to describe the fundamental characteristics of such an application. In particular, they allow to formalize the costs of asset creation. Another important property for such a DApp is the storage solution used for the asset images. Cryptokitties uses a centralized storage of NFT metadata, like most NFT management applications. However, this property is not exclusive to such applications, and our ontology proposes the formalization of the storage solution used for all DApps through the sbo:Storage class.

B. NFT MARKETPLACE

NFT Marketplaces, such as OpenSea [56], propose to their users the purchase and sale of NFT. Fig. 15 shows the user interface of OpenSea.

On this application, sellers are mostly companies that sell their NFTs, or users of the secondary market. On average, they will generate a large number of NFT creation and sale transactions on the platform.

On the other hand, buyers have a low number of transactions: few of them buy hundreds of different NFTs. It can then be interesting to differentiate in the modeling of the DApp the two types of users.

The DataProperties specific to NFT Marketplaces are as follows:

- UserBuyer: Properties related to an NFT Buyer

- NbTxPerUserBuyer: Number of transactions each user buyer needs to execute
  - NbTxPerUserBuyerOnMainchain: Number of transactions each user buyer needs to execute on the main blockchain
  - NbTxPerUserBuyerOnLayerTwo: Number of transactions each user buyer needs to execute on the Layer Two platform.
- NbMbPerUserBuyer: Amount of data each user buyer needs to store
  - NbMbPerUserBuyerOnMainchain: Amount of data each user buyer needs to store on the main blockchain
  - NbMbPerUserBuyerOnLayerTwo: Amount of data each user buyer needs to store on the Layer Two platform
  - NbMbPerUserBuyerOnStorage: Amount of data each user buyer needs to store on the storage service.

- UserSeller: Properties related to an NFT Seller
  - NbTxPerUserSeller: Number of transactions each user seller needs to execute
    - NbTxPerUserSellerOnMainchain: Number of transactions each user seller needs to execute on the main blockchain
    - NbTxPerUserSellerOnLayerTwo: Number of transactions each user seller needs to execute on the Layer Two platform.
  - NbMbPerUserSeller: Amount of data each user seller needs to store
    - NbMbPerUserSellerOnMainchain: Amount of data each user seller needs to store on the main blockchain
    - NbMbPerUserSellerOnLayerTwo: Amount of data each user seller needs to store on the Layer Two platform
    - NbMbPerUserSellerOnStorage: Amount of data each user seller needs to store on the storage service.

We use the same DataProperties that we defined in section III-E, relating to the transaction costs and data storage of the application. However, the two types of users will not use the application in the same way, so a duplication of the user properties is made to take these differences into account.

C. DECENTRALIZED AUTONOMOUS ORGANIZATIONS

DAOs [57], are DApps operating on the model of companies. Thus, like the shares of a company, a DAO has a token, which offers to its owners access rights to the governance of the DAO. This governance can concern the updating of the application, its properties, etc.

A well-known example of a DAO is MakerDAO, which manages the Dai [58] *stablecoin*. The token associated with the DAO is Maker, and Maker owners can vote to influence many parameters of the application. The different governance interactions on this DAO are presented in Fig. 16. We see

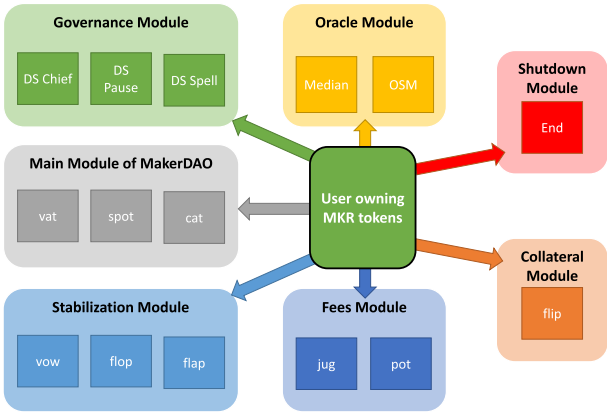


FIGURE 16. Example of a DAO, adapted from https://github.com/makerdao/dss/wiki#system-architecture. This figure shows the set of MakerDAO services that a user responsible for governance can influence.

in this diagram the whole of the modules of the DApp that the owners of the Maker (MKR) token can influence. These influences can be related to the classical functioning of the application, such as a modification of the loan rates or the stabilization of the protocol, but also on the critical functioning such as the shutdown of the system in case of a particular event like a fault or an abnormal behavior of the markets.

Finally, our ontology proposes to model DAOs in a similar way as NFT Marketplace, differentiating two types of users:

- Those who are responsible for the governance of the DAO,
- Those who simply use the services of the DAO.

The DAO-specific DataProperties are the following, for which we use the same nomenclature as before:

- UserForGovernance: Properties related to a user that participate in the governance of the DAO
  - NbTxPerUserForGovernance: Number of transactions each governance user needs to execute
    - NbTxPerUserForGovernanceOnMainchain: Number of transactions each governance user needs to execute on the main blockchain
    - NbTxPerUserForGovernanceOnLayerTwo: Number of transactions each governance user needs to execute on the Layer Two platform.
  - NbMbPerUserForGovernance: Amount of data each governance user needs to store
    - NbMbPerUserForGovernanceOnMainchain: Amount of data each governance user needs to store on the main blockchain
    - NbMbPerUserForGovernanceOnLayerTwo: Amount of data each governance user needs to store on the Layer Two platform
    - NbMbPerUserForGovernanceOnStorage: Amount of data each governance user needs to store on the storage service.
- TargetUser: Properties related to a target user of the DAO
  - NbTxPerTargetUser: Number of transactions each target user needs to execute

- NbTxPerTargetUserOnMainchain: Number of transactions each target user needs to execute on the main blockchain
- NbTxPerTargetUserOnLayerTwo: Number of transactions each target user needs to execute on the Layer Two platform.

- NbMbPerTargetUser: Amount of data each target user needs
  - NbMbPerTargetUserOnMainchain: Amount of data each target user needs to store on the main blockchain
  - NbMbPerTargetUserOnLayerTwo: Amount of data each target user needs to store on the Layer Two platform
  - NbMbPerTargetUserOnStorage: Amount of data each target user needs to store on the storage service.

D. ORACLES

Oracles are solutions for reading data from the outside world into a blockchain. To do this, DApp developers must define data sources that are considered secure. The values taken by these data sources can then be used as a source of truth.

The DAO MakerDAO, for example, requires for its operation that the DAO's smart contracts have access to the current price values of different cryptocurrencies. Indeed, MakerDAO users can recover Dai by placing different cryptocurrencies such as Ether as collateral. Collateral is a monetary value, in this case in cryptocurrency, that the user transfers to the contract. They will only be able to get this currency back if they repay the loan in Dai granted by the contract. In case the user does not repay the loan, then the contract keeps the collateral as compensation. The contract thus needs to know the value of the Ether to know how much Dai a user can receive from the value of the collateral they have placed. For this, a certain list of ETH price data sources are defined and managed by Omnia Relay. A consensus system for choosing the final value taken into account must also be defined. Liu *et al.* [59] specify the critical aspect of the received information: if the coalition of actors allowed to act on the oracle act maliciously, the currencies collateralized within the system become vulnerable. This example is specified in Fig. 17.

In this figure, several cryptocurrency exchanges provide real-time prices of different cryptocurrencies through their API. Then Omnia aggregates all these values, and provides them to MakerDAO contracts. Omnia's addresses are whitelisted to manage access to this functionality of the contracts. The smart contracts then calculate the final value of the prices they take into account.

Within our ontology, we can model the fact that a certain DApp uses an Oracle. However, this does not a priori impact the characteristics of the DApp we are studying, in terms of cost or latency.

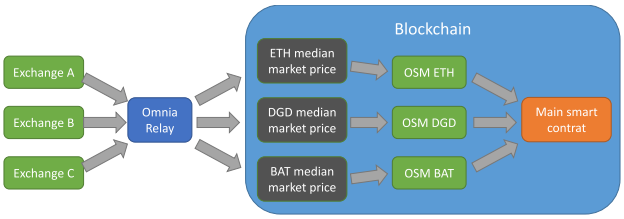


FIGURE 17. Example of an Oracle, inspired from https://docs.makerdao.com/smart-contract-modules/oracle-module.

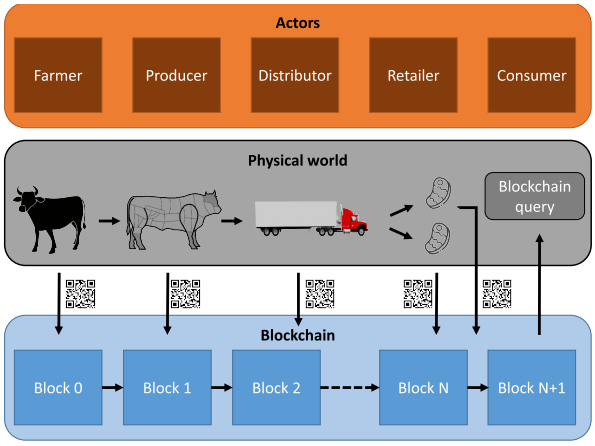


FIGURE 18. Example of an application of blockchain technology in supply chain, inspired from https://resolvesp.com/blockchains-supply-chains-part-ii.

E. SUPPLY CHAINS

The traceability of the transactions brought by the blockchain makes this technology relevant for the supply chain domain [60]. In particular, it is interesting to have a complete traceability of products, according to their references. Each reference corresponds to a family of identical products. For example, a furniture seller can propose in their catalog two references of tables and one reference of chair. Each of these references can then be produced in a certain number of units. The salesman could have 2000 different products: 250 tables of each reference and 1500 chairs.

Thus, for such DApps, a formalization centered on products and product references is relevant, and forms a blockchain pattern, presented in Fig. 18. This example shows a process of breeding and production of meat, in which all the actors involved in this process are users of the DApp. The different actors scan QR-codes allowing the data to be put on the blockchain in a simplified way for the users. At the end, the traceability of all products can be realized.

The DataProperties specific to supply chain DApps are the following, for which we use the same nomenclature as before:

- Product: Properties related to a product
  - NbTxPerProduct: Number of transactions for each product
    - NbTxPerProductOnMainchain: Number of transactions for each product on the main blockchain

    - NbTxPerProductOnLayerTwo: Number of transactions for each product on the Layer Two platform.
  - NbMbPerProduct: Amount of data to store for each product
    - NbMbPerProductOnMainchain: Amount of data to store for each product on the main blockchain
    - NbMbPerProductOnLayerTwo: Amount of data to store for each product on the Layer Two platform
    - NbMbPerProductOnStorage: Amount of data to store for each product on the storage service.
- Reference: Properties related to a specific product's reference in the supply chain DApp
  - NbTxPerReference: Number of transactions for each reference
    - NbTxPerReferenceOnMainchain: Number of transactions for each reference on the main blockchain
    - NbTxPerReferenceOnLayerTwo: Number of transactions for each reference on the Layer Two platform.
  - NbMbPerReference: Amount of data to store for each reference
    - NbMbPerReferenceOnMainchain: Amount of data to store for each reference on the main blockchain
    - NbMbPerReferenceOnLayerTwo: Amount of data to store for each reference on the Layer Two platform
    - NbMbPerReferenceOnStorage: Amount of data to store for each reference on the storage service.

In the case of blockchain Apps, we use the same pattern as defined in III-E. This pattern is illustrated by the NFT market place pattern.

However, here we do not differentiate according to the type of user, but according to the type of use of the application. A DApp with a single reference with many products will not be used in the same way as a DApp with many references in very limited quantities. Using the example of furniture sellers, the DApp of a seller offering 100000 products through a single reference will not have the same needs as a seller offering 50 products through 25 references. Indeed, in this last case, the DApp would need a way to search and filter the different references to make it easy for the buyer. The smart contracts developed need additional functionalities as a result.

F. NON-BLOCKING USER INTERFACES

A last blockchain pattern that we formalized corresponds to non-blocking user interfaces, proposed by Ojha [61]. With this type of user interfaces, a user can initiate several actions, without having to wait for the first action to be completed. This pattern is important in the blockchain domain, since this domain relies heavily on event-driven programming. This pattern describes that within a DApp, the user interface must not be blocking.

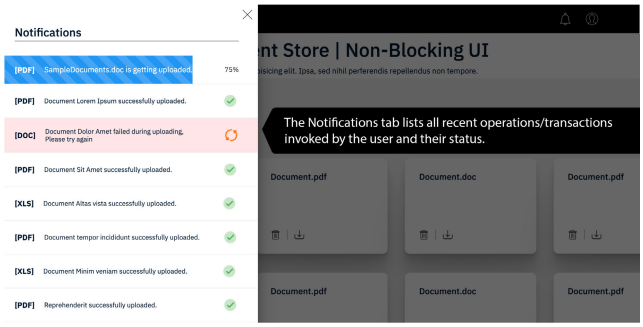


FIGURE 19. Example of a non-blocking user interface, taken from [61].

Fig. 19 shows an example of use of this blockchain pattern. It represents a website where a user can upload files in an interface, and use them. As uploading can take some time, a queue is visible on the left side of the figure. The user can interact with their uploaded files while others are still in the queue: adding a new file to the queue is non-blocking.

Concretely, for our ontology, we have chosen to differentiate the actions that can be performed simultaneously, and the actions that must be sequential. For example, for a DApp allowing the creation and sale of NFTs, a user who wishes to create one must upload an image file. Before the end of the file transfer, the user keeps the control of the application and can modify the description of the created NFT. These two actions can therefore be performed simultaneously. On the other hand, when another user wants to buy this NFT, they have to wait for the confirmation of the transaction before they can own it and use it, for example in a game. The purchase of the NFT is a blocking action for its use. However, the transaction can take several minutes before being validated. Thus, the application must leave the control to the user until the transaction is completed. Indeed, the user may decide to buy another NFT before the transaction purchasing the first one is completed.

The DataProperties specific to DApps implementing the blockchain pattern of non-blocking user interfaces are:

- ActionQueueSize: Total number of user actions available
- ActionCriticalQueueSize: Maximal number of sequential user actions.

G. SYNTHESIS ON BLOCKCHAIN PATTERNS

For each of these blockchain design patterns, we have defined DataProperties useful for their modeling. Thus, if we wish to model any application within our ontology, it is possible to use the DataProperties in order to create an adapted model. We did not need to add any new concepts or object properties. Indeed, the existing concepts and ObjectProperties are sufficient to formalize the blockchain patterns presented above.

It will then be necessary to write SWRL rules that allow to use these properties in order to deduce the operating characteristics of the concrete application. Indeed, it is difficult to write generic SWRL rules for each pattern, which do not need

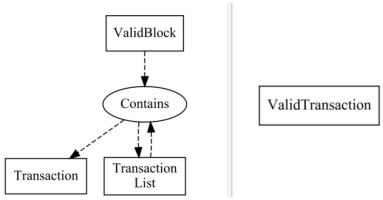


FIGURE 20. Graphical representation of rule 1. On the left, the antecedent of the rule. On the right, its consequent.

elements specific to the application to be modeled. However, the study of the SWRL rules present in the ontology allows the researchers to easily adapt them to their application, and thus to deduce an exhaustive modeling.

V. SEMANTIC WEB RULE LANGUAGE (SWRL)

A. SWRL AXIOMS DEFINITIONS

SWRL [62] is a language used to express rules acting on an OWL ontology. Each rule can be used by an inference engine to deduce ew axioms for the ontology. Then, new rules can be written in an iterative process.

SWRL rules follow a specific pattern. We define axioms that constrain individuals or their relations. These axioms can be part of the antecedent or the consequent of the rule. If each axiom of the antecedent is verified, then each axiom of the consequent will be verified as well.

1) RULES DEFINED FOR OUR ONTOLOGY

In the following, we describe 17 SWRL rules we defined for our blockchain ontology. First of all, the rules 1, 1bis and 2 define constraints on the main concepts of blockchain technology, and ensure our ontology formalizes the technology accurately.

Then, the rules 3 to 6 define constraints on the relation between the Ethereum blockchain and its sidechain Polygon. They let us understand how these two blockchains operate together, which improves their interoperability.

We then propose ten rules specific to DApps constraints, numbered A to D4. For example, they can deduce the cost and latency characteristics of a DApp, given its properties and the services used by the DApp.

a: RULE 1

A transaction included in the transaction list of a valid block is itself valid.

```
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 1. Rule 1.

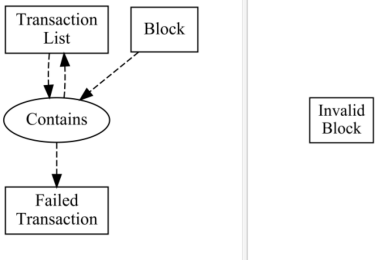


FIGURE 21. Graphical representation of the rule 1bis. On the left, the sequence of properties of the antecedent of the rule. On the right, the consequent of the rule.

The rule of listing 1 is simple. The lines 1 to 3 define the three individuals used: a transaction `?x`, a transaction list `?list`, and a valid block `?b`. Then, lines 4 and 5 define their inclusion: the `?b` block contains the list `?list`, which contains the transaction `?x`. Finally, line 7 defines the consequent of the rule, which is that the transaction `?x` is valid if all the axioms of the antecedent are valid.

The Protégé plugin called Aided OWL Notation (AOWLN) [63] gives a graphical representation of this rule, shown in Fig. 20. On the left, the antecedent of the rule. On the right, its consequent.

b: RULE 1BIS

A block that contains in its transaction list an invalid transaction is not valid.

```
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 2. Rule 1bis.

The listing rule 2 is thus complementary to the first rule, and is based on the same construction principle. Lines 1 to 3 define the three individuals used: a failed transaction `?x`, a transaction list `?list`, and a block `?b`. Then, lines 4 and 5 define their inclusion relations, and line 7 defines the consequence of the rule.

As before, we use the AOWLN plugin to obtain a graphical representation of this rule in Fig. 21.

c: RULE 2

A valid block on a blockchain using PoW is necessarily a block mined by a person on that same blockchain.

Like the previous rules, this second rule, defined in the listing 3, deals with the basic concepts of blockchains. The first four lines define the individuals used: a blockchain `?blockchain`, a valid block `?b`, the proof of work `?pow` and a blockchain account `?x`. Line 5 ensures that the

```
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 3. Rule 2.

considered blockchain works by PoW, the other consensus methods not having miners. Lines 6 and 7 define respectively the links "the blockchain `?blockchain` contains the block `?b`" and "the block `?b` is mined by the account `?x`." Finally, the last line concludes that `?x` mines for `?blockchain`.

d: RULE 3

If a Polygon transaction is referenced by its hash within a valid Ethereum transaction, then it is also valid.

```
1  gbo:PolygonTransaction(?x)  ∧
2  gbo:EthereumTransaction(?y)  ∧
3  gbo:ValidTransaction(?y)  ∧
4  gbo:HasHash(?x, ?h)  ∧
5  gbo:Contains(?y, ?h)
6  →
7  gbo:ValidTransaction(?x)
```

LISTING 4. Rule 3.

The rule in the listing 4, as well as the three following ones, deals with the links between the Ethereum blockchain and its Polygon sidechain. Here, we consider that a Polygon transaction is fully validated by Ethereum via a Checkpoint.

We define lines 1 to 3 a Polygon transaction, an Ethereum transaction that is also a valid transaction. Lines 4 and 5 state that the Ethereum transaction must contain the hash of the Polygon transaction. In this case, line 7 states that this Polygon transaction is valid.

e: RULE 4

If a Polygon transaction has existed for more than a week, then it is either valid or invalid. This means that the transaction is no longer awaiting validation by a challenge, since the time limit for someone to provide proof of the transaction's invalidity has passed.

The listing 5 defines a fourth rule, a little more complex than the previous ones. It relies on the fact that Polygon works through a system of *Fraud proof* as defined in the II-B1 section. Thus, if a transaction between Polygon and Ethereum is propagated by a node, the other nodes of the network have one week to contradict the given claims. Therefore, if a Polygon transaction is older than 7 days, then it is either validated or rejected by the network.

```
1  gbo:Polygon(?blockchain)  ∧
2  gbo:PolygonTransaction(?x)  ∧
3  gbo:TxSubmittedTime(?x, ?y)  ∧
4  gbo:Contains(?blockchain, ?x)  ∧
5  gbo:LastBlockTime(?blockchain, ?z)  ∧
6  swrlb:dayTimeDuration(
     ?dayTimeDurationLiteralWeek, 7, 0, 0,
      0)  ∧
7  swrlb:subtract(?dayTimeDurationVariable
     , ?z, ?y)  ∧
8  swrlb:greaterThan(
9      ?dayTimeDurationLiteralWeek,
10     ?dayTimeDurationVariable)
11 →
12 gbo:SettledTransaction(?x)
```

LISTING 5. Rule 4.

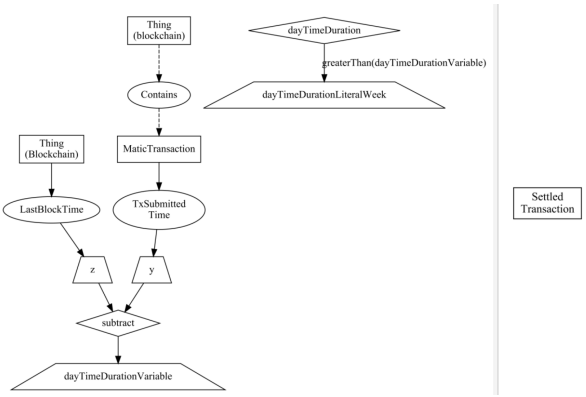


FIGURE 22. Graphical representation of the rule 4. On the left, the sequence of properties of the antecedent of the rule. On the right, the consequent of the rule.

Here, lines 1 to 4 define a Polygon transaction that is time-subjected and included in the Polygon blockchain. In lines 5 to 10, we use functions provided by SWRL, called *built-ins*, which allow time management. We compute the time difference between the transaction submission time `?x` and the time of the last Polygon block, `?z`. We then compare this gap to a time literal describing the delay of one week. Lines 8 to 12 indicate that if the time difference is greater than one week, then the transaction is either valid or invalid.

Again, AOWLN allows us to obtain Fig. 22. We see graphically that the rule involves more complex constraints than before.

f: RULE 5

The Polygon sidechain is periodically saved on the Ethereum mainchain (in the form of a *Merkel root* of new blocks).

Similarly to rule 4, we use the *built-ins* of SWRL time. The rule in the listing 6 proposes to include a Polygon checkpoint on the next block mined on Ethereum. However, one must be aware that this rule is not necessarily respected, but rather reflects the expected functioning of Polygon. Indeed, each

```
gbo:Polygon(?m) ∧
gbo:LastBlockTime(?m, ?y) ∧
gbo:LastCheckpointTime(?m, ?z) ∧
swrlb:dayTimeDuration(
  ?dayTimeDurationLiteralHalfHour, 0,
  0, 30,  0) ∧
swrlb:subtract(?dayTimeDurationVariable
  , ?z, ?y) ∧
swrlb:greaterThan(
    ?dayTimeDurationLiteralHalfHour,
    ?dayTimeDurationVariable) ∧
gbo:Ethereum(?eth) ∧
gbo:NextBlock(?eth, ?b) ∧
gbo:PolygonCheckpoint(?h)
→
gbo:Contains(?b, ?h)
```

LISTING 6. Rule 5.

miner chooses the transactions they include in their blocks. The miner of the next block can thus choose not to include the Polygon checkpoint, even if it has already been propagated through the network. Indeed, the users of a blockchain propagate desired transactions through the network: transactions waiting for validation are included in the *mempool*. However, nothing prevents a miner from not including a certain transaction in the created block.

We define lines 1 to 3 the Polygon blockchain, the date of its last block and the date of the last Checkpoint performed. Then, lines 4 to 8, we look if the difference between these two times is more than 30 minutes. In lines 9 to 11, we define the Ethereum blockchain, its next block, and a Polygon checkpoint. Finally, line 13 concludes that if the last Polygon checkpoint is more than 30 minutes old, then a new Checkpoint will be included in the next Ethereum block.

g: RULE 6

A Polygon transaction is included in a Polygon checkpoint if it is submitted between the date the Polygon transaction is submitted, but not later than 30 minutes.

```
gbo:PolygonTransaction(?tx) ∧
gbo:TxSubmittedTime(?tx, ?x) ∧
gbo:PolygonCheckpoint(?checkpoint) ∧
gbo:TxSubmittedTime(?checkpoint, ?y) ∧
temporal:add(?xPlusHalfHour, ?x, 30, "
   Minutes") ∧
temporal:before(?x, ?y) ∧
temporal:before(?y, ?xPlusHalfHour)
  →
gbo:IncludedIn(?x,?checkpoint) ∧
gbo:Contains(?checkpoint,?x)
```

LISTING 7. Rule 6.

The rule of the listing 7 allows to link a Polygon transaction to the Ethereum transaction in which it is included, by the checkpoint process. Lines 1 to 4 allow to define a Polygon transaction `?tx` submitted to a time `?x`, as well as a Polygon checkpoint `?checkpoint` submitted to a time `?y`. Then, we use the temporal time *built-ins* which allow some calculations on dates, in a more robust way than those included in SWRL. Lines 5 through 7 ensure that the transaction is older than the checkpoint, but not older than 30 minutes. If these constraints are not met, the transaction would be included within another checkpoint. Lines 9 and 10 define the object properties linking the Polygon transaction and its checkpoint: the transaction `?tx` is included in the checkpoint, and the checkpoint `?checkpoint` contains the transaction.

h: SPECIFIC RULE A

Specific Rules A to D4 allow to constrain the concept of DApp and their properties, according to the services used by the application.

The rule in the listing 8 allows to deduce the developer costs of a DApp, from the number of transactions to be performed, the amount of data to be stored, and the technologies used by the application.

```
sbo:DecentralizedBlockchainApplication(
  ?dba) ∧
gbo:Uses(?dba, ?blockchain) ∧
gbo:Blockchain(?blockchain) ∧
gbo:Uses(?dba, ?l2)  ∧
sbo:LayerTwo(?l2)  ∧
gbo:Uses(?dba, ?storage)  ∧
sbo:Storage(?storage)  ∧
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 8. Specific Rule A.

Rules A and B are relatively complex since they consist in the calculation of the costs associated with the considered DApp.

Lines 1 to 7 define a DApp and the technologies used by it: the blockchain used `?blockchain`, the transaction scalability solution `?l2`, and the storage solution `?storage`. If a DApp does not use one of these services, the individual will simply be a `NullLayerTwo` or a `NullStoage`, which does not interfere in the rest of the rule.

Then, lines 8 to 17 aim at defining the characteristics of the DApp, thus the number of transactions to be carried out by the developer and the quantity of data to be stored, as well as the properties of the blockchain services used: the costs of the transactions and the data storage. Lines 18 to 23 use this data to calculate the cost of the DApp, which is applied in property of the DApp in line 25. The calculation formula is as follows:

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

i: SPECIFIC RULE B

The listing 9 describes the specific rule B, which makes it possible to deduce the user costs of a DApp, starting from the number of transactions to be carried out, the quantity of data to be stored, and the technologies used by the application. This rule is therefore analogous to rule A, but for user and non-developer costs.

This rule reproduces exactly the same pattern as the previous one, but considers user costs instead of developers. Lines 1 to 7 define a DApp and the technologies used by it. Then, lines 8 to 17 aim at defining the characteristics of the DApp. Lines 18 to 23 use this data to calculate the cost of the DApp, which is applied in property of the DApp in line 25. The calculation formula is as follows:

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

j: SPECIFIC RULES C

The rules described below allow to deduce the latency (in reading and writing) of the transactions of a DApp.

Fig. 23 shows the graph obtained with AOWLN for the first of these rules.

The rule C1 of the listing 10 allows to deduce the read latency of the transactions of a DApp, when a transaction scalability service is used by the application.

```
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 9. Specific Rule B.

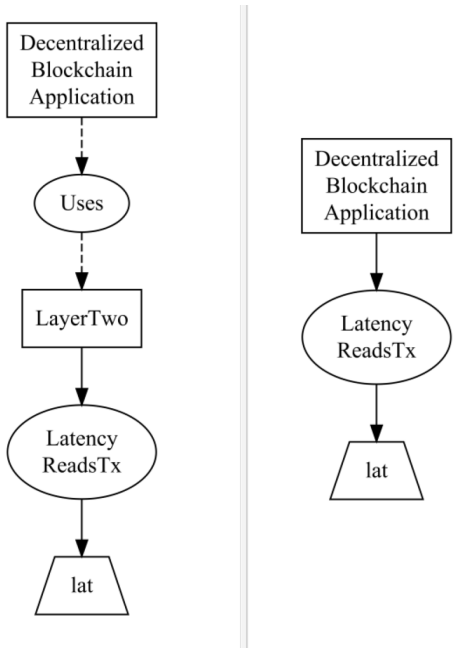


FIGURE 23. Graphical representation of the specific rule C1. On the left, the sequence of properties of the antecedent of the rule. On the right, the consequent of the rule.

The C2 rule of the listing 11 specifies that a DApp may not implement a scalability service. The class `sbo:NullLayerTwo` allows to manage this case easily.

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

LISTING 10. Specific Rule C1.

```
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 11. Specific Rule C2.

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

LISTING 12. Specific Rule C3.

The C3 rule, defined by the listing 12, allows to deduce the latency in writing transactions of a DApp, when a transaction scalability service is used by the application.

```
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 13. Specific Rule C4.

The rule C4 proposed in the listing 13 specifies that a DApp may not implement a scalability service. The class `sbo:NullLayerTwo` allows to manage this case easily.

k: SPECIFIC RULES D

The rules described below are used to deduce the latency (read and write) of a DApp's data storage.

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

LISTING 14. Specific Rule D1.

Rule D1 in the listing 14 is used to infer the read latency of a DApp's data storage, when a storage service is used by the application.

```
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 15. Specific Rule D2.

The D2 rule of the listing 15 specifies that a DApp may not implement a storage service. The class `sbo:NullStorage` allows to manage this case easily.

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

LISTING 16. Specific Rule D3.

The listing 16 introduces a new rule, D3, which is used to infer the write latency of a DApp's data storage, when a storage service is used by the application.

Similarly to rule D2 we specified a rule to infer the write latency regarding a DApps without a storage service, which is then managed with the class `sbo:NullStorage`.

2) OPEN WORLD ASSUMPTION ISSUES

OWL is a language supporting Open World Assumption (OWA). OWA means that OWL considers as potentially true everything that has not been defined in the ontology. This aspect in particular considerably limits the inferences that can be made about our case study. Indeed, the blockchain being a decentralized and distributed technology, it is impossible to know that a certain data does not exist. Concretely, for the

rule 5, we had to define the concept of CheckpointPolygon, but we cannot retrieve all the Polygon blocks over a period of time.

3) INFERENCES FOUND

Within the section III-D, we defined three individuals sbo:_DBA1, sbo:_DBA2 and sbo:_DBA3. For each of these individuals, we determined some DataProperties values after modeling the applications. We also determined some typical values of transaction and storage costs for the technologies used.

With this information and the rules described above, we were able to use the Drools [64] inference engine to derive new axioms for our ontology. We chose this reasoner since it is directly integrated with the SWRLTab [65] Protege plugin used for our SWRL rules. The reasoner must first translate the OWL ontology and the SWRL rules into Drools, then run the inference engine, and finally translate the result into OWL axioms, which are then automatically added to the ontology. All these three tasks are executed in a negligible time of 750ms, and produce 469 new axioms, including 261 axioms related to data or object properties. Examples of inferred axioms are presented in the listing 17. This first axiom defines the latency in writing of the individual sbo:_DBA1. The second axiom defines that the checkpoint sbo:_PolygonCheckpoint2 contains the transaction sbo:_PolygonTransaction2. We detail this second axiom in the section V-B.

```
1 sbo:_DBA1 sbo:LatencyWrites "14.0"^^xsd:
    double
2 sbo:_PolygonCheckpoint2 gbo:Contains sbo:
    _PolygonTransaction2
```

LISTING 17. Inferences found from DApp-specific SWRL rules.

The inductions found by the inference engine are detailed in the listing 18 and in the table 3.

For the first two cases (lines 1 and 3 of listing 18), the developer costs are the same, 2€ per user according to our assumptions. However, the user costs are higher for the first application case (lines 2 and 4 of listing 18). Indeed, users perform transactions on Ethereum in this first case, whereas they perform them on Polygon for the second. The user costs are therefore 10€ for the first case, and only 0.01€ for the second case.

The third application case also uses Polygon, so the user costs are the same as for the second DApp. However, using Polygon as a blockchain and not just as an Ethereum sidechain also reduces the developer costs from 2€ per user to only 0.002€.

The results obtained are analyzed in the next section.

4) ANALYSIS OF THE RESULTS

First, we see that our ontology allows us to differentiate the characteristics of the three modeled DApps. From these results, we can better understand the impact that the different potential technologies have on the properties of the application.

```
1 sbo:CostDevPerUser(_dba1, 2.0)
2 sbo:CostUser(_dba1, 10)
3 sbo:CostDevPerUser(_dba2, 2.0)
4 sbo:CostUser(_dba2, 0.01)
5 sbo:CostDevPerUser(_dba3, 0.002)
6 sbo:CostUser(_dba3, 0.01)
```

LISTING 18. Inferences found from DApp-specific SWRL rules.

TABLE 3. Characteristics of the 3 defined DApps found by Drools, in average euro.

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

We see that the differences in costs, both for developers and users of the DApp, have a very significant variance depending on the technology used to develop the application. Here we see that:

- The costs of an Ethereum-only solution are considerable, both for developers and users.
- Using Polygon in conjunction with Ethereum reduces user costs considerably. Developer costs are not reduced, since they perform all their transactions on Ethereum. This solution is well suited if the DApp does not have too many users.
- The use of Polygon alone does not change the user costs, compared to the second case considered. However, the developer costs are greatly reduced.

One of the limitations of the results obtained is the non-inclusion of the constraints related to the security of the DApp. Indeed, we could think that we should then choose to use only Polygon, as the main blockchain, for this use case, since the costs are reduced. However, Polygon uses a consensus method, DPoS, which has a weaker intrinsic security than Ethereum's PoW. When Polygon is used as a sidechain of Ethereum, this disadvantage is reduced, since Ethereum provides some of the security of the DApp. Thus, because of this limitation, we would not recommend using Polygon alone. If the developer costs are prohibitive, we would recommend using other developer transaction scalability solutions, such as *rollups*.

B. MAPPING BETWEEN TWO BLOCKCHAIN PLATFORMS

An application of the written SWRL rules is the mapping between two blockchain platforms, in our case Ethereum and Polygon.

The ontology presented in section III allows, among other things, to formalize various blockchain services, including specific blockchain instances. Within this ontology, we were

able to model the concepts related to Ethereum and those related to Polygon.

We then constrained the concepts of the ontology by SWRL rules. In particular, the rules Rule 3 to Rule 6 allow to constrain the concepts related to Ethereum, Polygon, and their links. Indeed, the Rule 5 indicates that every 30 minutes, a new Polygon checkpoint is created on Ethereum, through an Ethereum transaction. The Rule 3 indicates that the Polygon transactions that are included in this checkpoint are validated, and the Rule 6 links the Polygon transactions to their checkpoint. Finally, the Rule 4 describes the operation of the one-week challenge period linking Polygon and Ethereum.

The objective is then to run the Drools reasoner on the ontology and the defined SWRL rules, in order to obtain elements of comparison between Polygon and Ethereum. Indeed, understanding the differences between the properties of, for example, a Polygon transaction and an Ethereum transaction allows a DApps developer to know the consequences of a platform change for his application.

As mentioned before, the inference engine has allowed to define some axioms linking Ethereum to Polygon. In particular, the Rule 6, allowing to match a Polygon transaction with the checkpoint that includes it, allows to find the inferences detailed in the listing 19. Thus, the checkpoint _PolygonCheckpoint1 contains the first transaction, and the checkpoint _PolygonCheckpoint2 contains the other two transactions.

This is the expected result, since _PolygonCheckpoint1, submitted to Ethereum on 18/10/2021 at 18:30, validates the Polygon transactions submitted that same day between 18:00 and 18:30, which is the case for _PolygonTransaction1. Similarly, _PolygonCheckpoint2 validates Polygon transactions submitted between 6:30pm and 7:00pm, which is the case for _PolygonTransaction2 and _PolygonTransaction3.

```
gbo:Contains(_PolygonCheckpoint1,
    _PolygonTransaction1)
gbo:IncludedIn(_PolygonTransaction1,
    _PolygonCheckpoint1)
gbo:Contains(_PolygonCheckpoint2,
    _PolygonTransaction2)
gbo:IncludedIn(_PolygonTransaction2,
    _PolygonCheckpoint2)
gbo:Contains(_PolygonCheckpoint2,
    _PolygonTransaction3)
gbo:IncludedIn(_PolygonTransaction3,
    _PolygonCheckpoint2)
```

LISTING 19. Inferences found from rule 6.

Thus, we have found inferences to link the concepts of Ethereum and Polygon, which improves the interoperability between these two blockchains.

In the application section of our research, we will detail the concrete use of this mapping.

C. CONSTRAINING BLOCKCHAIN CONCEPTS

When a new concept is formalized within our ontology, we had to constrain it. To do this, a first step is to consider the relations with the other defined classes. With OWL, we can link two classes by ObjectProperties. In our case, as blockchains can be seen as particular data structures, several of its relations are inclusion or content relations.

If we wish to constrain the data associated with a class that has no direct link with another existing class, we can use DataProperties. These properties allow one to declare certain properties associated with a class.

Sometimes, the OWL language is not expressive enough to constrain concepts sufficiently. In this case, we use SWRL rules to constrain several concepts, the ObjectProperties, and the DataProperties that link them.

D. SYNTHESIS

We specify SWRL rules to improve the expressiveness of the ontology described in section III. Indeed, SWRL allows to obtain constraints between the concepts of the ontology more simply and efficiently than using only OWL. For example, when running the SWRL rules defined earlier, 469 additional axioms are added to the ontology that would have been difficult to manually determine.

Our specification of SWRL rules have three objectives:

- Formalize fundamental constraints related to blockchains, such as the validity rules of blocks and transactions of a blockchain
- Formalize the links between the Ethereum blockchain and its sidechain Polygon
- Formalize the properties of DApps, and in particular the costs and latency of these applications.

Other advantages of the defined SWRL rules are their scalability and modularity. Indeed, it is easy for other researchers or developers to formalize their applications by adapting these rules or by adding new ones. For example, rules analogous to the specific rules for DApps A and B can be written to formalize the cost constraints of DApps that use blockchain patterns as described in section IV.

Finally, SWRL has other applications in the blockchain domain. Choudhury *et al.* [66] indeed proposes the use of SWRL rules in order to automatically generate smart contracts from documents detailing the regulations of any domain. This kind of application can also help the development of DApps, and our rules allow to support their formalization.

VI. APPLICATION IN THE VIDEO GAME INDUSTRY

A. MOTIVATIONS

Our goal is to apply our research and use the ontology previously designed in a concrete industrial example. As a result, in the following, we apply our research to the blockchain video game industry. In particular, we use the ontology to formalize B2Expand's game LTR.

LTR is a multiplayer real time game.

B. APPLICATION OF OUR RESEARCH

Thanks to our ontology, we can describe in more detail the technical differences in the transition from one of the two blockchains to the other. We have formalized the two blockchains, as well as some rules constraining the functioning of Ethereum and Polygon. We could deduce that using Polygon as a scalability solution for Ethereum was the best solution for our application.

We thus modeled the different concepts of Ethereum and Polygon within our ontology. We also wrote the SWRL rules corresponding to the interactions between these two blockchains, described in Rule 3 to Rule 6.

These rules constrain the concepts of Polygon and Ethereum transactions, and include for example the concepts of checkpoints and transaction hashes.

We were able to infer correspondences between Polygon transactions and the Polygon checkpoints that validate these transactions. These mappings provide insight into the semantic differences between Polygon and Ethereum. On the other hand, we did not obtain any other inferences to better define these differences. This may mean that we did not constrain the modeled concepts enough, or that the relationships between these concepts are too broad to be quantified without using too many intermediate concepts. Indeed, the objective is not to manually model the differences between the two blockchains, but to obtain them from reasoning on simple rules.

Nevertheless, this ontology allows us to deduce other constraints. A simple example of such a constraint is that the users need to have a blockchain Polygon account. Fortunately, obtaining this account is straightforward since it follows the same standards as Ethereum. Thus, every Ethereum user automatically has an associated Polygon account.

Finally, to use Polygon within LTR, our methodology proposes to proceed as follows:

1) Within the game developed with Unity, replace all calls to the Ethereum blockchain with calls to the Polygon blockchain. This works since Polygon is based on the same ecosystem as Ethereum.
2) Similarly, on the NFTs management DApp developed with web technologies, it is possible to ask users to connect to Polygon with the same tools they usually use for Ethereum.
3) The mint of the assets, realized by B2Expand, is realized only on Ethereum, so this part does not change of functioning.
4) However, the transfer of assets between players will now be done on Polygon, reducing the associated costs.

Fig. 20 shows a Business Process Model and Notation (BPMN) model describing an example of an LTR asset lifecycle using the Polygon solution as an Ethereum sidechain.

Here, B2Expand creates and mints the asset only on Ethereum, then transfers it to Polygon's master contract via an Ethereum transaction. User 1 then buys the asset on Polygon, and can transfer it with very low costs to user 2. User 2 wants to recover the asset on Ethereum. Indeed, it is possible that they want to use this asset on a platform that does not support the use of Polygon. They then request the withdrawal of the asset on Ethereum and wait for the challenge period.

VII. EVALUATION OF THE PROPOSED BLOCKCHAIN ONTOLOGY

A. EVALUATION CRITERIA OF THE BLOCKCHAIN ONTOLOGY

1) EVOLUTIVITY

As the field of blockchain evolves rapidly, it is essential to ensure that the contributions of our work do not become obsolete with the arrival of a new technology. The scalability of our contributions consists in the elaboration of a protocol providing for their updates according to the changes that the blockchain ecosystem undergoes.

For our ontology, scalability consists in ensuring that the formalization of new concepts remains possible, without considerably impacting the concepts already modeled.

2) EASE OF USE OF THE ONTOLOGY

Similarly, it should be easy to extend an ontology if one wants it to be used. Ontology experts can be supported with guidelines to the formalization of certain common concepts in DApps.

3) SYNTHESIS

The table 4 summarizes the different evaluation criteria chosen.

TABLE 4. Criteria for evaluating our ontology.

Criteria	Operationalization
Scalability	Resilience to the appearance of new blockchain services
Ease of use ontology	knowledge level to have in order to formalize a new DApp

B. RESULTS OF THE EVALUATION OF THE BLOCKCHAIN ONTOLOGY

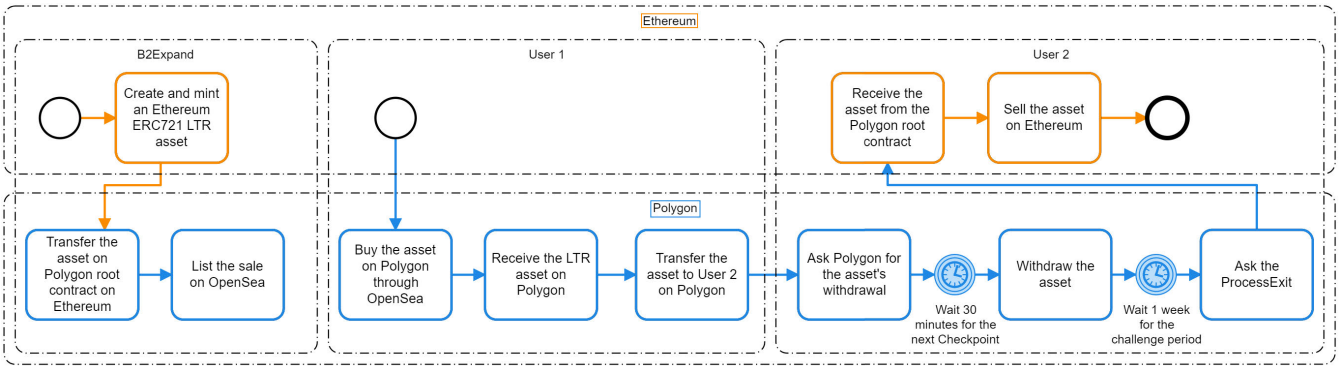
1) EVOLUTIVITY

Our ontology is scalable for several reasons. First, if a new type of blockchain service appears, we can associate latency and cost properties to it like other services. The modeling is modular: we can define which components form a given DApp. Other SWRL rules can also be written if one needs to formalize more complex constraints on a particular blockchain service.

In addition, other researchers can add concepts, or extend our ontology with new blockchain ontologies. Adding concepts related to the economic issues of different actors could be considered.

2) EASE OF USE OF THE ONTOLOGY

As mentioned before, we have integrated partial formalizations of common blockchain patterns. This makes it easy to use the ontology to formalize new use cases of it.



LISTING 20. **BPMN showing the transfer of LTR assets between Ethereum and Polygon.**

To formalize more complex constraints, expertise in SWRL is necessary. It is possible to use existing rules within the ontology to write new ones, but the expressiveness of this language can make this task difficult for complex problems.

The table 5 summarizes the results for the criteria we considered.

TABLE 5. **Results obtained on the evaluation of our research contributions.**

Criteria	Results
Evolutivity	The ontology is evolutive
Ease of use ontology	The ontology guides users with blockchain patterns

VIII. EVALUATION OF THE INDUSTRIAL APPLICATION

A. EVALUATION CRITERIA FOR THE INDUSTRIAL APPLICATION

1) EVOLUTIVITY

The blockchain technology and the associated ecosystems are very innovative. This leads to a common difficulty when developing DApps: new technical solutions appear regularly, which can make the developments obsolete.

One of the evaluation criteria of our industrial application is therefore scalability. We have to make sure that the adopted solution cannot be made obsolete quickly without having the means to adapt it.

2) MODULARITY

The modularity of an application refers to its architecture. The maintenance of a modular application is easier than for a non-modular application. An architecture by microservices was particularly adapted for DApps.

Concretely, our application will have a good modularity if it is possible to adapt the different components of the application easily. This concept is therefore complementary to scalability.

3) USE OF THE ONTOLOGY

As the industrial application is used to validate our contributions, it is important that we use the developed ontology to the maximum. This allows us to study the possibilities that our ontology offers in terms of DApp formalization, but also its limitations.

4) SATISFACTION OF FUNCTIONAL CONSTRAINTS

The main objective of our methodology is to translate the specifications of a DApp into a list of service instances to be used to build the application. Thus, we have to evaluate the degree of satisfaction of the functional constraints present in the initial specifications.

5) SATISFACTION OF TECHNOLOGICAL CONSTRAINTS

Another objective of the methodology is the satisfaction of the technological constraints obtained from the BPMN modelling and the ontology constructed.

Concretely, if this evaluation criterion is not fully satisfied, it means that the final application will not work correctly, since the constraints of scalability, latency or service integration will not be well respected.

6) USER EVALUATION

A final interesting criterion for our evaluation is to take into account the opinions of experts in blockchain game development and users of our game. This will allow us to compare the results obtained with what they think is the correct way to develop such a DApp, and to make sure that the user experience is ergonomic.

7) SYNTHESIS

The table 6 summarizes the different evaluation criteria chosen.

B. RESULTS FOR THE INDUSTRIAL APPLICATION

1) EVOLUTIVITY

Our industrial application depends on the blockchain on two main points. First, the LTR game, developed with the help of the Unity game engine, must have read access to the information contained in the deployed smart contracts. Indeed, we need to know, for each player, which graphical assets they have access to. The connection to the blockchain

TABLE 6. Evaluation criteria for our industrial application.

Criteria	Operationalization
Evolutivity	Resilience to new blockchain services
Modularity	Possibility of updating a functionality without modifying the others
Usage of ontology	Degree of formalization by ontology
Functional constraints	Number of constraints taken into account
Technological constraints	Number of constraints taken into account
User evaluation	N.A.

is done by the Nethereum *package* for Unity, specific to the Ethereum ecosystem. Thus, we depend on this ecosystem for the LTR game. However, an important part of blockchain projects being developed today remain compatible with this ecosystem.

A second point concerns the application to manage the NFT of a player. This application needs a write access to the blockchain. Indeed, each player can connect to his Metamask account to perform transactions on our DApp. Here, we also depend on the Ethereum ecosystem, but in a less important way. Indeed, the integration with web technologies is a priority of the new projects. Thus, it will be easy to make the transition to another solution if needed. We will only need to call different APIs.

As for the transition process between Ethereum and Polygon, our application must be scalable in order to adapt to the changes of the different third party projects involved. Indeed, Ethereum and Polygon protocols are still under development at the moment. Ethereum is aiming for the transition to PoS during 2022, and Polygon is developing an ecosystem conducive to multi-chain. For this, in addition to the elements detailed in the previous point, the application being modular, we can develop test modules calling the APIs of the protocols under development. Once the specifications are fixed, we can then validate the proper functioning of the new protocols. This point is further detailed in the paragraph VIII-B2c Modularity of the blockchain platforms used.

2) MODULARITY

a: MODULARITY OF SMART CONTRACTS

The smart contracts for managing LTR assets are partly modular. Indeed, the proxy design pattern allows one to change certain functionalities of the application by deploying again only the smart contract managing this functionality. The other contracts are then not impacted by this change. An example of a possible application of this feature is the change of the asset creation cost formula. The creation of LTR assets requires the burning of the B2Expand cryptocurrency, the Nexium. We can adjust the number of Nexium to be burned depending on the crypto-economic conjuncture at a certain time, i.e. the prices of the crypto-currency markets.

b: MODULARITY OF USER INTERFACES

The set of user interfaces supporting the blockchain allows to use all the required functionality: calling a specific smart contract, reading data on the blockchain, connecting to the account of a blockchain, managing different networks, etc.

So if we develop a new game that doesn't use Unity like LTR does, the new game engine will be able to reuse all the other features of our applications. Of course, this works on the condition that it is possible to connect to the blockchain with this new engine. That's why the different user interfaces are modular.

c: MODULARITY OF THE BLOCKCHAIN PLATFORMS USED

As mentioned earlier, the tools for connecting to the blockchain support different protocols. Thus, the connection tools to Ethereum or to Polygon are the same. The modularity of the blockchain platforms used thus comes down to the modularity of the configuration files used, as well as the different technical constraints.

For example, if we were to move from Polygon to another platform, we will have to make sure, thanks to our methodology, that this new platform is well adapted to the constraints of the application. If it is the case, the transition to a new Layer Two is done in a similar way to the transition from Ethereum to Polygon.

3) USE OF THE ONTOLOGY

The ontology in our research contributions allow us, among other things, to formalize the transition from Ethereum to Polygon for our DApp. Detailing the different concepts related to these two platforms then allows us to ensure the relevance of the written programs.

We also used SWRL rules related to cost and latency constraints to analyze the three possible use cases of our application. The use of SWRL allows a great expressiveness of these constraints.

4) SATISFACTION OF FUNCTIONAL CONSTRAINTS

The basic functional constraint of the developed application is the management of NFT for the LTR game. This constraint is fully validated, since all the standards allowing this management are implemented. Moreover, we have added some specific functionalities to our DApp, such as the burning of Nexium or the grouping of assets.

The transition to the use of Polygon also allows us to prepare the satisfaction of new functional constraints. For example, it will be possible to perform writes to the blockchain during game-related events, such as a player's victory or the collection of items in the game that could give the player a corresponding NFT. However, these new features are not yet implemented within the game, so it is not possible for us to fully evaluate them.

5) SATISFACTION OF TECHNOLOGICAL CONSTRAINTS

For the satisfaction of the functional constraints, we have carried out integration tests to validate the fact that the use of Polygon does not pose any particular problems for the management of our assets. These tests show no difficulties in replacing the Ethereum APIs by Polygon APIs for the in-game blockchain queries.

6) USER EVALUATION

Once a new version of the LTR game incorporating the new functionalities brought by our work will be released, we will then be able to collect feedback from users and our industrial partners on our DApp.

7) SYNTHESIS

The table 7 synthesizes the different results obtained.

TABLE 7. Results obtained on the evaluation of our industrial application.

Criteria	Results
Evolutivity	The DApp is evolutive
Modularity	The DApp is modular on three axes: smart contracts, user interfaces, and the blockchain platforms used
Usage ontology	The formalizations of the three cases (using only Ethereum, only Polygon, or Polygon as a sidechain of Ethereum) allowed us to choose which case to implement
Functional constraints	The functional constraints of NFT management are validated. The functional constraints linked to the use of Polygon will be validated once the LTR game new version is released.
Technological constraints	The technological constraints linked to the management of NFT on Polygon are validated. The writing on the blockchain from the game will be validated once the game is released.
User evaluation	The NFT management solution for LTR will be open so that all users can test the application

IX. CONCLUSION AND PERSPECTIVES

We built a DApps blockchain ontology to formalize DApps. Existing ontologies are modeling blockchain systems and therefore do not have the same goal as our DApps ontology. In particular, our DApps ontology can aid the development of DApps by modeling blockchain services that can be used in a DApp.

We applied and validated our contributions in the Video Game industry, particularly with the LTR game, and wrote SWRL rules to constrain the concepts of a DApp. Through the definition of blockchain patterns, this application of our DApps ontology can be easily generalized to many use cases.

As a perspective, additional SWRL rules can be added in order to add new constraints. For example, an SWRL rule could translate the concepts related to different blockchain systems. Another way to aid the interoperability between multiple blockchain systems would be by extending our ontology to formalize current solutions to cross-chain transfers.

APPENDIX
ADDITIONAL INDIVIDUALS OF THE ONTOLOGY

This appendix gives additional classes in our ontology. We have not included them in III-B for sake of simplicity.

8) GBO:LAYERTWO

a: SBO:_MULTICHAIN

This individual represents a scalability solution in multichain.

b: SBO:_ROLLUP

This individual represents a scalability solution via rollups.

9) GBO:STORAGE

a: SBO:_CENTRALIZEDSTORAGE

This individual represents a centralized storage, such as a classic website.

b: SBO:_IPFS

This individual represents IPFS distributed storage.

c: SBO:_NULLSTORAGE

This individual allows to consider a DApp which does not have a dedicated storage.

d: SBO:_P2PSTORAGE

This individual represents peer-to-peer distributed storage. This means that DApp users share DApp files directly with each other.

10) GBO:ENGINE

a: SBO:_UNITY

This individual represents a front-end created by the Unity game engine. It is for example useful for the B2Expand company Light Trail Rush (LTR) game, since this game needs to call smart contracts on Ethereum from an executable created with Unity.

b: SBO:_SPA

This individual represents a front-end created with web technologies such as React or Angular. These are SPAs. This example is useful to model our Non-Fungible Token (NFT) management DApp developed for the B2Expand LTR game. As explained earlier, NFTs are representations on the blockchain of game assets. Players can buy, sell or trade them freely. The possession of an NFT is then reflected in the game.

REFERENCES

[1] Z. Zheng, S. Xie, H.-N. Dai, X. Chen, and H. Wang, "Blockchain challenges and opportunities: A survey," *Int. J. Web Grid Services*, vol. 14, no. 4, pp. 352–375, 2018.

[2] Y. Guo and C. Liang, "Blockchain application and outlook in the banking industry," *Financial Innov.*, vol. 2, no. 1, pp. 1–12, Dec. 2016.

[3] A. Azaria, A. Ekblaw, T. Vieira, and A. Lippman, "MedRec: Using blockchain for medical data access and permission management," in *Proc. 2nd Int. Conf. Open Big Data (OBD)*, Aug. 2016, pp. 25–30.

[4] T. Min, H. Wang, Y. Guo, and W. Cai, "Blockchain games: A survey," in *Proc. IEEE Conf. Games (CoG)*, Aug. 2019, pp. 1–8.

[5] R. Belchior, A. Vasconcelos, S. Guerreiro, and M. Correia, "A survey on blockchain interoperability: Past, present, and future trends," *ACM Comput. Surveys*, vol. 54, no. 8, pp. 1–41, Nov. 2022.

[6] L. Besancon, C. F. D. Silva, and P. Ghodous, "Towards blockchain interoperability: Improving video games data exchange," in *Proc. IEEE Int. Conf. Blockchain Cryptocurrency (ICBC)*, May 2019, pp. 81–85.

[7] J. Pfeffer, "EthOnIntroducing semantic Ethereum," ConsenSys Media, New York, NY, USA, Tech. Rep., 2017. Accessed: May 8, 2022. [Online]. Available: https://media.consensys.net/ethon-introducing-semantic-ethereum-15f1f0696986

[8] N. Six, N. Herbaut, and C. Salinesi, "Blockchain software patterns for the design of decentralized applications: A systematic literature review," *Blockchain, Res. Appl.*, vol. 3, no. 2, Jun. 2022, Art. no. 100061.

[9] 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 *Proc. ACM SIGSAC Conf. Comput. Commun. Secur.*, Oct. 2016, pp. 3–16.

[10] *Proof of Authority: Consensus Model. With Identity at Stake*, Poa Network, San Francisco, CA, USA, 2017.

[11] B. Magri, C. Matt, J. B. Nielsen, and D. Tschudi, "Afgjort—A semi synchronous finality layer for blockchains," Cryptol. ePrint Arch., Tech. Rep. 2019/504, 2019, p. 44. Accessed: May 8, 2022. [Online]. Available: https://eprint.iacr.org/2019/504

[12] E. Buchman, "Tendermint: Byzantine fault tolerance in the age of blockchains," Ph.D. dissertation, Dept. School Eng., Univ. Guelph, Guelph, ON, Canada, 2016.

[13] K. Chatterjee, A. K. Goharshady, and Y. Velner, "Quantitative analysis of smart contracts," in *Proc. Eur. Symp. Program.*, Cham, Switzerland: Springer, 2018, pp. 739–767.

[14] U.-R. Hector and C.-L. Boris, "BLONDiE: Blockchain ontology with dynamic extensibility," 2020, *arXiv:2008.09518*.

[15] D. L. McGuinness and F. Van Harmelen, "Owl web ontology language overview," *W3C Recommendation*, vol. 10, no. 10, p. 2004, 2004.

[16] *Everdreamsoft/Sandra*, EverdreamSoft, Geneva, Switzerland, Sep. 2019.

[17] S. Delgado-Segura, C. Pérez-Sola, G. Navarro-Arribas, and J. Herrera-Joancomartí, "Analysis of the bitcoin UTXO set," in *Proc. Int. Conf. Financial Cryptogr. Data Secur.*, Cham, Switzerland: Springer, 2018, pp. 78–91.

[18] C. Sguanci, R. Spatafora, and A. M. Vergani, "Layer 2 blockchain scaling: A survey," 2021, *arXiv:2107.10881*.

[19] K. Karantias, "SoK: A taxonomy of cryptocurrency wallets," Cryptol. ePrint Arch., Tech. Rep. 2020/868, 2020. Accessed: May 8, 2022. [Online]. Available: https://eprint.iacr.org/2020/868

[20] W. Warren and A. Bandeali. (2017). *0x: An Open Protocol for Decentralized Exchange on the Ethereum Blockchain*. [Online]. Available: https://github.com/0xProject/whitepaper

[21] J. Poon and T. Dryja, "The bitcoin lightning network: Scalable off-chain instant payments," Tech. Rep., 2016. Accessed: May 8, 2022. [Online]. Available: https://lightning.network/lightning-network-paper.pdf

[22] V. Buterin, "An incomplete guide to rollups," Tech. Rep., 2021. Accessed: May 8, 2022. [Online]. Available: https://vitalik.ca/general/2021/01/05/rollup.html

[23] B. DiFrancesco, "Validium and the layer 2 two-by-two—Issue No 99," Build Blockchain Tech, Tech. Rep. 99, 2020. Accessed: May 8, 2022. [Online]. Available: https://www.buildblockchain.tech/newsletter/issues/no-99-validium-and-the-layer-2-two-by-two

[24] *Blockchain Solutions for Gaming. Whitepaper V2 Draft*, Funfair Technologies, Dublin, Ireland, 2018. Accessed: May 8, 2022. [Online]. Available: https://funfair.io/wp-content/uploads/FunFair-Commercial-White-Paper-v2-draft.pdf

[25] J. Coleman, L. Horne, and L. Xuanji, "Counterfactual: Generalized state channels," L4 Res., Tech Rep., 2018. Accessed: May 8, 2022. [Online]. Available: https://l4.ventures/papers/statechannels.pdf

[26] *Loopring Protocol Design*, Loopring, Shanghai, China, Sep. 2021. Accessed: May 8, 2022. [Online]. Available: https://github.com/Loopring/protocols/blob/master/packages/loopring_v3/DESIGN.md

[27] *Overview|Zksync: Secure, Scalable Crypto Payments*, Matter Labs, George Town, Cayman Islands, Sep. 2021. Accessed: May 8, 2022. [Online]. Available: https://docs.zksync.io/userdocs/intro.html

[28] *Offchain Labs Dev Center*, Inside Arbitrum, New York, NY, USA, 2022. Accessed: May 8, 2022. [Online]. Available: https://developer.offchainlabs.com/docs/inside_arbitrum

[29] *Introduction| StarkEx V3*, StarkWare Industries, Netanya, Israel, Sep. 2021. Accessed:May 8, 2022. [Online]. Available: https://docs.starkware.co/starkex-v4/

[30] *Deversifi Javascript Trading API*, Deversifi, Tel Aviv, Israel, Jan. 2022. Accessed: May 8, 2022. [Online]. Available: https://docs.deversifi.com/

[31] A. Back, "Enabling blockchain innovations with pegged sidechains," Blockstream, San Francisco, CA, USA, Tech. Rep., 2014. Accessed: May 8, 2022. Available: https://blockstream.com/sidechains.pdf

[32] D Larimer, "DPOS consensus algorithm—The missing white paper—Steemit," Tech. Rep., Nov. 2019. Accessed: May 8, 2022. [Online]. Available: https://steemit.com/dpos/@dantheman/dpos-consensus-algorithm-this-missing-white-paper

[33] S. King and S. Nadal, "PPCoin: Peer-to-Peer crypto-currency with proof-of-stake," *Self-Published Paper*. Accessed: May 8, 2022. [Online]. Available: https://decred.org/research/king2012.pdf

[34] J. Poon and V. Buterin, "Plasma: Scalable autonomous smart contracts," Tech. Rep., 2017. Accessed: May 8, 2022. [Online]. Available: https://plasma.io/plasma-deprecated.pdf

[35] *Loom Network—The Next-Generation Blockchain Application Platform for Ethereum*, Loom Network, Seoul, South Korea, 2022. Accessed: May 8, 2022. [Online]. Available: https://loomx.io/

[36] *Polygon Ethereum's Internet of Blockchains*, Polygon, North Andover, MA, USA, Feb. 2021.

[37] *Polygon|Ethereum's Internet of Blockchains*, Polygon, Bengaluru, India, 2022. Accessed: May 8, 2022. [Online]. Available: https://polygon.technology/

[38] J. Kanani, S. Nailwal, and A. Arjun, "Matic whitepaper," Polygon, Bengaluru, India, Tech. Rep., Sep. 2021. Accessed: Sep. 16, 2021. [Online]. Available: https://github.com/maticnetwork/whitepaper

[39] G. Wood, "Polkadot: Vision for a heterogeneous multi-chain framework," Parity Technologies, London, U.K., White Paper 21, 2016. Accessed: May 8, 2022. [Online]. Available: https://polkadot.network/PolkaDotPaper.pdf

[40] J. Kwon and E. Buchman, "Cosmos whitepaper, Cosmos, Zug, Switzerland, Tech. Rep., 2019. Accessed: May 8, 2022. [Online]. Available: https://v1.cosmos.network/resources/whitepaper

[41] (2022). *Harmony—Scaling Ethereum Applications and Cross-Chain Finance*. Harmony, Mountain View, CA, USA. Accessed: May 8, 2022. [Online]. Available: https://blog.harmony.one/harmony-keynote-scaling-ethereum-applications-cross-chain-finance/

[42] J. Benet, "IPFS–content addressed, versioned, P2P file system," 2014, *arXiv:1407.3561*.

[43] M. Zichichi, S. Ferretti, and G. D'Angelo, "On the efficiency of decentralized file storage for personal information management systems," in *Proc. IEEE Symp. Comput. Commun. (ISCC)*, Jul. 2020, pp. 1–6.

[44] *A Decentralized Storage Network for Humanity's Most Important Information*, Filecoin, Palo Alto, CA, USA, 2017. Accessed: May 8, 2022. [Online]. Available: https://filecoin.io/

[45] S. Wilkinson, T. Boshevski, J. Brandoff, and V. Buterin, *Storj a Peer-to-Peer Cloud Storage Network*. Atlanta, GA, USA: Storj Labs, 2014.

[46] R. Materese, "Blockchain," Nat. Inst. Standards Technol., Gaithersburg, MD, USA, Tech. Rep., Sep. 2019. Accessed: May 8, 2022. [Online]. Available: https://www.nist.gov/blockchain

[47] (Mar. 2022). *Checkpoint|Polygon Technology|Documentation*. Polygon, Bengaluru, India. Accessed: May 8, 2022. [Online]. Available: https://docs.polygon.technology/docs/contribute/heimdall/checkpoint/

[48] L. Lamport, R. Shostak, and M. Pease, "The Byzantine generals problem," in *Concurrency, Works Leslie Lamport*. New York, NY, USA: Association for Computing Machinery, 2019, pp. 203–226.

[49] N. Szabo, "Formalizing and securing relationships on public networks," *First Monday*, vol. 2, no. 9, pp. 1–21, Sep. 1997.

[50] V. Buterin, "A next-generation smart contract and decentralized application platform," Tech. Rep., 2014. Accessed: May 8, 2022. [Online]. Available: https://blockchainlab.com/pdf/Ethereum_white_paper-a_next_generation_smart_contract_and_decentralized_application_platform-vitalik-buterin.pdf

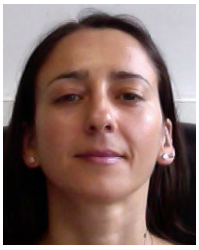
[51] E. Androulaki, A. Barger, V. Bortnikov, C. Cachin, K. Christidis, A. D. Caro, D. Enyeart, C. Ferris, G. Laventman, Y. Manevich, and S. Muralidharan, "Hyperledger fabric: A distributed operating system for permissioned blockchains," in *Proc. 13th EuroSys Conf.*, Apr. 2018, pp. 1–15.

[52] N. E. Ioini and C. Pahl, "A review of distributed ledger technologies," in *Proc. Move Meaningful Internet Syst. (OTM) Conf.*, H. Panetto, C. Debruyne, H. A. Proper, C. A. Ardagna, D. Roman, and R. Meersman, Eds. Cham, Switzerland: Springer, 2018, pp. 277–288.

[53] X. Xu, I. Weber, and M. Staples, "Blockchain patterns," in *Architecture for Blockchain Applications*. Cham, Switzerland: Springer, 2019, pp. 113–148.
[54] (Sep. 2021). *Mintable.App Home*. Mintable, Singapore. Accessed: May 8, 2022. [Online]. Available: https://mintable.app/
[55] (2022). *Cryptokitties. Cryptokitties | Collect and Breed Digital Cats*, Dapper Labs, Vancouver, BC, Canada. Accessed: May 8, 2022. [Online]. Available: https://www.cryptokitties.co/
[56] *OpenSea, the Largest NFT Marketplace*, OpenSea, New York, NY, USA, Sep. 2021.
[57] C. Jentzsch, "Decentralized autonomous organization to automate governance," Slock.it, Berlin, Germany, White Paper, Nov. 2016. Accessed: May 8, 2022. [Online]. Available: https://lawofthe level.lexblogplatformthree.com/wp-content/uploads/sites/187/2017/07/WhitePaper-1.pdf
[58] *The Dai Stablecoin System*, Maker, Loretto, KY, USA, 2017.
[59] B. Liu, P. Szalachowski, and J. Zhou, "A first look into DeFi oracles," 2020, *arXiv:2005.04377*.
[60] G. M. Hastig and M. S. Sodhi, "Blockchain for supply chain traceability: Bus. requirements and critical success factors," *Prod. Operations Manage.*, vol. 29, no. 4, pp. 935–954, Apr. 2020.
[61] V. Ojha. (2020). *GitHub—IBM/BlockchainDevelopmentDesignPatterns*. IBM, Armonk, NY, USA. Accessed: May 8, 2022. [Online]. Available: https://github.com/IBM/BlockchainDevelopmentDesignPatterns
[62] I. Horrocks, "SWRL: A semantic web rule language combining OWL and RuleML," *W3C Member Submission*, vol. 21, no. 79, pp. 1–31, 2004.
[63] J. Nguyen, J. Geyer, T. Farrenkopf, and M. Guckert, "Aided OWL notation (AOWLN): Conceptual modelling and visualisation of advanced SWRL rules," in *Proc. 10th Int. Joint Conf. Knowl. Discovery, Knowl. Eng. Knowl. Manage.*, 2018, pp. 175–182.
[64] M. Proctor, "Drools: A rule engine for complex event processing," in *Proc. Int. Symp. Appl. Graph Transformations With Ind. Relevance*. Cham, Switzerland: Springer, 2011, p. 2.
[65] (Oct. 2021). *Home Protegeproject/Swrlapi Wiki*. Protégé Project, Stanford, CA, USA. Accessed: May 8, 2022. [Online]. Available: https://github.com/protegeproject/swrlapi
[66] 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," in *Proc. IEEE Blockchain Initiative*, Sep. 2018, pp. 1–5.
[67] L. Besançon, P. Ghodous, J.-P. Gelas, and C. F. D. Silva, "Modelling of decentralised blockchain applications development," in *Proc. Int. Conf. High Perform. Comput. Simulation (HPCS)*, Barcelone, Spain, Mar. 2021, pp. 1–7.

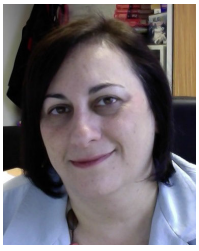
LÉO BESANÇON received the degree in computer science engineering from the École Centrale de Lyon, in 2018, the master's degree in artificial intelligence from Université Claude Bernard Lyon 1, in 2018, and the Ph.D. degree in computer science from the LIRIS Laboratory, Université Claude Bernard Lyon 1, as well as the company B2Expand, in 2021, focused on blockchain systems interoperability.

He has published a poster paper for the 2019 IEEE International Conference on Blockchain and Cryptocurrencies and a conference paper for the 2020 International Conference on High Performance Computing and Simulation.



CATARINA FERREIRA DA SILVA received the Ph.D. degree, in 2007, and the Habilitation degree in information science and technology, in 2020.

She is an Associate Professor and the Director of the master program in management of information systems with the Department of Information Science and Technology, University Institute of Lisbon (ISCTE), Portugal. She is a member of the Expert Panel of the European Blockchain Observatory and Forum, which is an initiative sponsored by the European Commission and the Directorate-General of communications networks, content and technology. She had the pleasure of co-supervise five defended Ph.D. students and has been involved in several European funded projects, such as Blockchain for ICT Professionals, VET4APPS, Keystone, Nebula, Trust, Towntology, Tempus Mitcon, CONNIE, FUNSIEC, and SPICE. She has published several journals and conference papers in prestigious venues, such as IEEE International Conference on Blockchain and Cryptocurrency, *Telematics and Informatics Journal*, the *International Journal of Agile Systems and Management*, *Computers in Industry*, and *Journal of Intelligent Manufacturing*. She is a member of the Program Committee of the Track Decentralized Applications (DAPP) with Blockchain, DLT and Crypto-Currencies of the ACM Symposium on Applied Computing and the European, Mediterranean and Middle Eastern Conference on Information Systems.



PARISA GHODOUS is currently a Full Professor with the Computer Science Department, University of Lyon I; and a member of the Laboratory of Computer Graphics, Images and Information Systems (LIRIS UMR 5205). She was involved in more than 20 European projects, four as a Coordinator, such as VET4APPS, BLISS, MACHINA, and CHAISE. Her research interests include blockchain, cloud computing, interoperability, web semantic, service science, collaborative modeling, product data exchange, and modeling and standards.

Prof. Ghodous is in editorial boards of *CERA*, *ICAE*, and *IJAM* journals; and in the committees of many relevant international associations, such as concurrent engineering and interoperability.

JEAN-PATRICK GELAS received the graduate degree (D.E.A.) in computer science from the École Normale Supérieure de Lyon (ENS), Lyon, France, in 2000, and the Ph.D. degree from Universite Claude Bernard Lyon 1, France, in December 2003.

He was awarded the National Grant from the French Ministry of Research and Technology (MENRT), he pursued his doctorate research/work at INRIA Team (called RESO) of the LIP Laboratory, ENS Lyon. He has spent the next year in USA, holding a postdoctoral (a Research Associate) position at the Logistical Computing and Internetworking (LoCI) Laboratory, Computer Science Department, The University of Tennessee. He is currently an Assistant Professor, teaching computer science (system, computer networks, embedded system, and blockchain technologies) at the Université Claude Bernard Lyon 1. His research interests include large scale distributed systems, like blockchain technologies.