Modelling of Decentralised Blockchain Applications Development

Léo Besançon, Parisa Ghodous, Jean-Patrick Gelas, Catarina Ferreira da Silva

Modelling of Decentralised Blockchain Applications Development

Léo Besançon, Parisa Ghodous, Jean-Patrick Gelas
Univ Lyon, Université Claude Bernard Lyon 1
LIRIS, F-69100
Villeurbanne, France
{leo.besancon, parisa.ghodous}@liris.cnrs.fr
jean-patrick.gelas@univ-lyon1.fr

Catarina Ferreira Da Silva
Instituto Universitário de Lisboa (ISCTE-IUL)
Lisboa, Portugal
catarina.ferreira.silva@iscte-iul.pt

***Abstract*—The development of Decentralised Blockchain Applications (DBA) is becoming more and more complex. We formalise the development of DBA based on Business Process Modelling and ontologies. This formalisation permits to represent a set of specific constraints that characterise a DBA. These constraints concern data management, the Blockchain storage cost, the number of transactions allowed in a given time and the execution of the application logic. Based on this formalism, we propose a generic and adaptive methodology for the development of DBAs. This methodology uses as input the constraints related to each DBA and as output recommends a set of appropriate services necessary for the development of each DBA. For example, our methodology can recommend a specific distributed storage service such as InterPlanetary File System (IPFS) for a DBA that needs to handle a large amount of data. We apply our methodology to a Blockchain Video Game application.**

Keywords—Blockchain; Business Process Models; Ontologies; Video Games; Development; Decentralised Blockchain Applications

I. INTRODUCTION

Decentralised Blockchain Applications (DBA), otherwise called Dapps are applications that integrate Blockchain technology to achieve various goals: decentralisation, no single point of failure, automation, trustless transactions, privacy and/or traceability [1]. However, a Blockchain Application may not be decentralised if part of the application is centralised. As such, integrating a Blockchain in an application does not mean this application becomes decentralised.

We find that the topic of architectures for DBA is not sufficiently developed. Indeed, various proposals aiming to formalise generic architectures for distributed applications have been proposed, e.g. by Microsoft [2] and IBM [3], but application-specific constraints are not yet considered.

In order to fulfil this gap, we propose a methodology improve modelling and formalisation of specific DBAs.

We then apply our methodology in the domain of Video Games. We define what a Blockchain Video Game is and what the constraints related to them are. Then, we formalise a Blockchain architecture suitable for such games.

As the field of Blockchain technology is still relatively new, our research is motivated by the need for a semantic and formal approach to DBA development. Kim and Laskowski [4] state that the use of ontology-driven development has two advantages. Firstly, we get a better understanding of the data handled in Blockchain systems, so it leads to better data standards and business practices. Secondly, it helps to create formal specifications for automated inference and verification of Blockchain-based processes.

A. What are Decentralised Blockchain Applications?

We define Decentralised Blockchain Applications as applications that use Blockchains in order to take advantage of one or several of their properties and characteristics. This definition is coherent with the one proposed by Raval [5], stating that DBAs should be open source, have internal cryptocurrency support, a decentralised consensus and no central point of failure.

One of the most important properties of this definition is the criterion of decentralisation, which is the shared governance of an application's state. As stated in [6], decentralisation is not a binary property. Depending on the number of actors involved and the distribution of power in the shared governance, an application can have various degrees of decentralisation.

In fact, as shown in [7], most current instances of DBAs have to make compromises regarding decentralisation. For example, some DBA can choose to compromise decentralisation for the scalability of the number of supported transactions. An example of such a compromise can be seen with Azarus [8]. The company behind this DBA uses the EOS [9] Blockchain for their application, which supports a huge number of transactions. However, the Delegated Proof of Stake [10] consensus mechanism behind this Blockchain is more centralised than the Proof of Work [11] or Proof of Stake [12] consensus mechanisms.

Currently developed DBAs are more and more complex, and span various application fields, from decentralised finance (or DeFi) [13] and [14] or resource trading [15] to video games [16]. The video game industry can use Blockchain technology to ensure the players' asset integrity over time, reduce server costs for developers, as well as to allow easy trading of in-game assets for currency.

As the level of complexity of the DBAs increases, a few research works have been formalising what is needed to develop

Supported by organisation B2Expand, 69100 Villeurbanne, FRANCE.

a DBA. For example, Duan et. al. [17] formalise Blockchains as a standalone service. In practice, Blockchains interact with various other systems such as user interfaces, game engines and distributed storage systems. However, a Blockchain standalone service is insufficient for our goals.

Abdellatif and Brousmiche [18] produced a representation model of both Blockchains and users of a DBA to allow for formal verification of smart contracts. This work is more related to the security of a DBA than to its development.

Another type of formalisation is through the design of a generic architecture for a DBA [3]. These types of formalisation do not take sufficiently into account the constraints of a specific use-case. For example, having a distributed file storage component in an architecture is helpful, but it does not tell a developer what specific distributed file storage solution is best suited for his/her target application.

B. The need for a generic methodology for Decentralised Blockchain Applications development

One key aspect of DBAs is that for more advanced use cases, such applications cannot solely rely on a Blockchain. Indeed, Blockchains bring multiple constraints in application design.

Firstly, data sharing through Blockchain has high latency. This constraint can be viewed in two distinct ways.

- If we only need data to be propagated in the network without the Blockchain immutability and security, the Blockchain will not add any latency on top of the distributed data transfers. In this case, the latency is higher than in a centralised system but is both suitable for most applications and able to be accurately modelled [19].
- However, if we need the data to be immutable, then we need to wait for a certain number of Block confirmations in order for the probability of a rollback to become negligible. Here, the confirmation time depends on the selected consensus algorithm, as, for instance, Proof of Work (PoW) [11] confirms blocks slower than Proof of Stake (PoS) [12] or Delegated Proof of Stake (DPoS) [10].

Data sharing is also affected by the network utilisation, and other parameters such as Block size, and finally the security one needs.

Blockchain also has low data sharing bandwidth. This means that Blockchains are neither suitable for use cases needing a high transaction count, nor for use cases needing to handle a large amount of data.

One particularly unsuitable use case for Blockchains should thus be video games, as multiplayer video games usually need low latency to be in real time, high communication rate between players, and large image files to be transferred. This use case will be detailed in section IV, but it shows the need to include other components in DBAs design.

To resolve this data sharing problem, a DBA developer can choose to design his/her application according to design patterns that can be grouped into two main categories:

- The use of semi-centralised systems to avoid using the Blockchain technology for tasks that are not suitable for Blockchains. For example, most existing DBAs, such as CryptoKitties [20], have their front-end code stored on centralised web pages. In the case of CryptoKitties, the images of the virtual assets are served by the servers of the company, causing a single point of failure. Although the definition of DBAs implies they rely only on decentralised systems, this compromise solution seems to be accepted by the community. Unfortunately, this single point of failure may lead to attacks such as DNS Hijacking [21].
- The use of existing decentralised scaling solutions. For example, it is possible to store a Single Page Application (SPA) on a distributed storage solution such as InterPlanetary File System (IPFS) [22]. One example of DBA that uses IPFS to achieve a fully decentralised game is made by EtherPlay [23]. We describe these solutions in more details in section III.B.

These usual approaches to the design of DBAs have proven to work, but the resulting architectures are rarely suitable for multiple applications. This leads to low interoperability between existing DBAs, which decreases their usability. In our opinion, there is a strong need to develop a methodology that, given a specific use case for a Blockchain application, leads to a DBA architecture suitable for this application. In this paper, firstly we study the state of the art of Blockchain systems architectures.

After studying how we could improve them, we propose a novel approach and procedure to formalise DBAs. Our approach uses ontologies and Business Process Model and Notation (BPMN) to gather the constraints for the application and define the set of service instances needed for the DBA. As a result, we obtain a specified architecture for the DBA. Then, we show how we apply our methodology in the Blockchain video game industry. As far as we know no other existing system proposes a methodology similar to ours.

II. CURRENT STATE OF THE ART OF BLOCKCHAIN APPLICATION ARCHITECTURES

A. Concepts related to architectures in service oriented computing

One possible approach to design an architecture is to compose the application in services that each performs one feature of the application. We can differentiate two main approaches in current architecture design: Service Oriented Architectures (SOA) and Microservice Architectures.

According to [24], Microservices are more suitable for decentralised workflows. Indeed, they emphasise the difference between service orchestration and service choreography. Service orchestration happens where a central element synchronises the different services of an application. On the contrary, service choreography relies on the fact that a service can synchronise with the services it interacts with directly. As a result, an architecture for DBAs can essentially rely upon an approach based on Microservices.

B. Architecture of Blockchains and Distributed Ledger Technologies (DLTs)

Traditional databases have a centralised and permissioned governance. A user is assigned a role and access level, e.g. administrator, write access, read access, etc.

On the contrary, Blockchains, as well as other types of DLTs often either have a decentralised and permissionless governance, or permissioned but distributed governance. In this case, the term distributed refers to multiple machines performing a calculation together, whereas decentralised refers to the control and governance of the calculation. Permissionless and permissioned refers to the access rights of each machine to participate in the given calculation.

Cai et. al. [25] propose a generic architecture for Blockchain systems. Their architecture is fairly simple, with three components: the Blockchain as a data structure, a Peer to Peer network, and a consensus model.

Different Blockchains and DLTs can use various consensus mechanisms and have different protocol parameters. This means they each have different characteristics, in terms of features, scaling of the number of transactions they handle, data bandwidth, latency, etc.

For the sake of simplification, we choose not to take into consideration these characteristics in our methodology. Instead, we choose to focus our work on the characteristics of the other services, such as distributed storage, user interfaces, or distributed computation services.

C. Architecture of Decentralised Blockchain Applications

One of the major works for DBA architecture in the industry has been designed by IBM [3]. They describe an architecture for DBAs that is suitable for enterprise applications. However, it only describes high-level systems. As such, they propose an exhaustive view of what features each component of a DBA should have, but not how to best select these features.

Other works on DBA architectures have a more application-oriented approach. For example, [26] specifies a reference architecture for Blockchain-based Peer-to-Peer IoT applications. They target only one type of use case, but they focus on providing concrete solutions for developers of IoT applications who want to integrate Blockchain technology. Indeed, they focus on handling payments and identity for a huge number of devices. They do not consider applications that require to store large amounts of data on the blockchain, recommending using other communication channels such as UDP or TCP in order to do so. Furthermore, they also recommend using permissioned blockchains for real time IoT applications, as the use of public blockchains leads to latency issues. These architectures make it easier to design and develop a DBA. They describe what services will be needed to be implemented for a DBA and how to handle interactions between them.

However, we find some limitations in the use of these architectures to entirely design a DBA. For example, a DBA developer would use one of the architectures described previously in order to know what services he/she should implement for his/her application. However, he/she will not be able to deduce what concrete solutions should be implemented.

D. Formalisation tools for DBAs

We use formalisation tools described in this section in order to propose a methodology that helps to formalise the concrete constraints of a DBA and recommend a suitable set of solutions to implement.

1) Ontologies

Firstly, ontologies are useful to describe Blockchain systems. Indeed, ontologies aim to semantically define the various concepts needed in a given field. Blockchain systems often involve different fields that use similar concepts, which may not have consistent definitions. As a result, several works are trying to semantically define what a Blockchain is.

For example, Blockchain Ontology with Dynamic Extensibility (BLONDiE) [27] and the Ethereum Ontology (EthOn) [28] use Web Ontology Language (OWL) describing such ontologies. They are useful to get a global understanding of how different Blockchain concepts such as transactions, address and signatures relate to one another, as well as to formalise these concepts, but we did not find any application using these ontologies. However, the PHP framework Sandra [29] lets users easily design their Blockchain ontologies, and is used by EverdreamSoft to query Blockchain assets.

Through the use of ontologies, the various systems interacting with and within a DBA all have the same definition of the concepts and data structures used, which improves semantic interoperability of a DBA.

2) Business Process Model and Notation

Ontologies help formally define the concepts and data structures we need to support in the application. Nonetheless, for an application that needs the users to interact with each other, another important aspect is how each possible action will be propagated inside the application and inside the network of users.

The best way to model this information propagation is through BPMN [30] modelling. Indeed, Business Process Models help formalise business workflows and processes. These methods are widely used in order to better understand the interactions between the various entities that need to communicate in an information system to accomplish given tasks. As a result, the modelling of all the processes comprised in a target Blockchain application shows what interactions to focus on during the design of the application.

3) Formalisation of video games

In 2015, Solís-Martínez et. al. [31] built VideoGame Process Modelling, a notation based on BPMN to model video game logic. Their aim is to quickly design video games with their notation, which can then be quickly converted into functional code. As such, their proposal mainly formalises the development aspect of video games. This formalisation is important, even though Politowski et al. [32] showed that video game projects are mainly programmed using the same processes as other computer software. However, the VideoGame Process Modelling formalisation gives us a better understanding of the interactions between the different services within a video game. This is important when we consider the formalisation of Blockchain video games.

III. PROPOSED METHODOLOGY

Based on a description of the target DBA, our methodology recommends a set of services needed in the DBA's architecture, and tells the user by which service each interaction in the DBA will be handled.

A. Formalisation of the needs of a DBA

In order to be able to recommend the needed services for the DBA, our methodology needs a formal description of the target DBA's features and its constraints.

This formal description is partly obtained through an ontology defining the needed concepts related to Blockchains, the field of the target use case, and the target DBA.

Another form of formalisation we consider is to obtain models using BPMN to show the various interactions the users will have with the application and with each other, as well as the interactions between the different services within the application.

1) Ontologies related to the field of the use case

In most cases, DBAs that are being designed integrate Blockchain technology inside an already existing application. Examples of existing DBAs are implementations of lotteries, gambling games, asset trading or exchanges for various finance derivatives. All of these applications existed before Blockchain technology, but we saw the benefits of building DBAs for these use cases.

However, in order to be sure that important concepts related to the field of the use case are translated correctly into the Blockchain ecosystem, it is useful to define an ontology covering all of these concepts. For example, if a developer wants to design a decentralised exchange for financial assets, he has to correctly identify and define formally what an asset is and what a trade between users of an exchange is. Constraints related to the security of a user's funds also need to be defined formally, regarding the login process, the number of Blockchain confirmations needed to assume a transfer is immutable, and so on.

Answering these questions will be needed for making various design choices that will be reflected in the DBA's architecture. For example, one could want to make an offer on this exchange without any fees, and to include fees only when a trade is executed. To achieve this, the DBA needs to include a way for users to sign messages that define a sale or buy order, and share the signed message with other users without committing a transaction on the Blockchain. This means the architecture has to be designed with this feature in mind, for instance through the use of a gossip protocol.

2) Ontologies related to the target Decentralised Blockchain Application

Our methodology we describe in section III and its application in the video game industry in section IV extends an existing Blockchain ontology in order to accurately define how concepts related to our use case's field, the chess game, are represented in a Blockchain setting.

To highlight the need of an extended ontology, we use the concept of *finality* [33], or immutability. One of the properties of Blockchains is to be immutable. This means that once consensus is reached and data is committed to a Blockchain, it should not be possible for this data to be edited or deleted. However, some Blockchains, such as the ones using the Proof of Work consensus mechanism, have probabilistic finality. This implies that once a Block is committed to the Blockchain, it may still be deleted, but with a probability that quickly decreases to zero with time.

A DBA should formally handle these edge cases, and this is easier to consider if we have a unified ontology that defines how each data of the application is specified with regards to the underlying Blockchain data structure.

3) BPMN model of the DBA

Another model we need to establish is an event-based BPMN model of the DBA itself. Here, we define the properties needed for each interaction within the DBA. This means this BPMN model shows the constraints of each interaction in terms of the number of transactions it should handle, the needed security, or the storage space needed to handle data transfers. As [34] shows, it is possible to extend BPMN in order to consider the decentralised environment of Blockchain systems.

Once a target DBA has been formalised, the developer should be able to easily match its needs with existing distributed services, such as the ones described in the next section.

B. Distributed services

As we have seen in existing DBA architectures, such as [3], several additional services are used to handle the tasks Blockchain cannot handle effectively. One of the goals of our methodology is to recommend which services are needed for the DBA, and for which interactions they are used. Indeed, if a Blockchain is suitable to be used for a given interaction, we recommend using it.

1) Distributed storage

For storing the ledger data, Blockchain currently requires every node to store it, even though solutions regarding sharding [35] and scaling [36] are being worked on. As a result, storing relatively small data for the DBA directly in the Blockchain can be quite expensive. Thankfully, other distributed storage solutions, like IPFS, Swarm, or direct Peer to Peer exchange between the users of the DBA are possible.

We propose in [37] a data representation for Blockchain Video Games assets that makes it easier to support several distributed storage solutions concurrently. For example, a user of a given DBA could query data simultaneously on IPFS and through Peer to Peer connections with other users of the DBA.

2) Distributed computations

The service for distributed computation is divided into two categories: scaling up the number of transactions a Blockchain can handle in a given time, or increasing the complexity of the computations.

Regarding the first category, scaling up the number of transactions, for instance Ethereum currently only handles a dozen transactions per second, which is not enough if one plans on using this Blockchain for a DBA that needs to handle more transaction in a given time period. Second layer scaling solutions

can be comprised of several methods to increase the number of transactions: mainly sharding, plasma, and state channels.

Each of these solutions have their own set of compromise between the complexity of implementation, the security of the transactions, the possible censorship, the decentralisation and the number of transactions they can handle.

Regarding the second category, increasing the complexity of computations that can be handled by a DBA may also be wanted. For example, a forward pass of a trained neural network would not be easily doable on Ethereum. This means that a DBA that requires distributed or decentralised executions of such computations would not be able to do so using a Blockchain.

Several distributed computing services are currently working towards this goal: iExec, Trubit, and Golem. These companies develop a service rendering possible to have trusted off-chain complex computations. Anybody could use their application to compute a certain workload. Then, they validate, for example through redundancy, the result of the computation, which is transmitted back to the user who offered this workload.

C. Goal of our methodology

Our methodology intends to make a set of recommendations to help the development of a DBA. For instance, for each interaction between users of the DBA, a recommendation is provided on whether it is considered best to only use a Blockchain to implement a certain DBA functionality or if it is needed to include other services. Our methodology starts with a list of functional requirements [38] for the application from the client, from which the specifications are drawn (Fig. 1).

These specifications are currently written in natural language, but we expect to be able to formalise them later.

From a security perspective, our methodology also needs to ensure the security benefits of Blockchain technology transfers well to the whole application. This is achieved by focusing our models on the possible interactions between all the actors of a given DBA.

D. Overview of our methodology

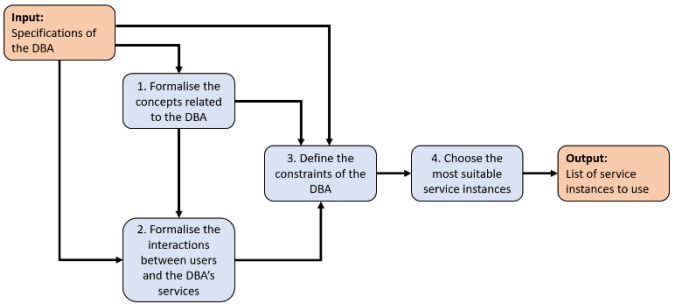

Figure 1: Overview of the four steps of our methodology.

Given a target DBA, the methodology is divided into the following steps, presented in Fig. 1:

1. Formalise both the concepts related to the Blockchain side of the DBA and those related to the application field. We can reuse an existing ontology or, if needed, we can extend an existing ontology to comprehensively define the concepts of our DBA. For instance, in our application presented in section IV, we extend the existing EthOn [28] ontology with concepts related to the chess game.
2. Formalise all interactions that happen in the DBA, between the services of the DBA or the users. We use event-based BPMN modelling for this step: the start event corresponds to an action a user or service makes within the DBA. It is for example the deployment of the DBA, or a transaction made by a user. Then, our model shows all the communications between users and services that ensue from this starting event. A DBA may need several different BPMN models in order to formalise every possible interaction.
3. Define the technical constraints of the DBA. Here, both the ontology and the BPMN models are useful to reach a comprehensive list of these constraints. Indeed, the ontology can help identify the constraints related to one concept in particular. For instance, if the ontology shows that the concept of Blockchain account is critical to the application, the account model of the Blockchain used could lead to a technical constraint. An Unspent Transaction Output (UTXO) model, which requires every transaction, aside from coinbase transactions that compensate miners, to refer to an existing transaction in order to be valid, could be preferred over a simpler account/balance model, which only keeps track of the state associated with each account [39]. Similarly, to define the constraints related to the interactions between the multiple concepts of the DBA, each interaction shown in the BPMN models has to be analysed.
4. Choose the suitable service instances for the DBA. For each constraint defined in the previous step, we choose the best service instance to answer it using the following criteria: security of the solution, decentralisation requirements, number of transactions to handle, volume of data to exchange, or even simplicity of implementation. The output of our methodology is obtained by matching the characteristics of each interaction in a DBA with the functionality of the services supported by our methodology. As an example, we apply our methodology to a DBA that is a registrar for images and their metadata. In this case, the costs involved prevent the storage of images on the Blockchain, so we use IPFS for this task. The access permissions for the DBA is handled by smart contracts on the Ethereum Blockchain, as the application does not require a high-throughput and low-latency execution environment. Finally, the user interface for storing and retrieving images is a small SPA, interacting with an Ethereum smart contract using Web3 and a Web3 provider such as Metamask.

IV. Application to the Video Game Industry

A. Modelling of a Blockchain chess game

In order to apply our methodology to the Video Game industry, we start by gathering the characteristics of video games and Blockchain video games.

Inspired by the work of Solís-Martínez et. al. [31], we propose the BPMN modelling of a two users video game, described in Fig. 2. In this BPMN model, we define a player of a game as an entity who can make actions that will affect the state of the game. The list of possible actions given a specific game state is constrained by the game's rules. For example, in a game of chess, the actions for a player include the moves that each piece could do, considering the current board position. Also, as chess is a turn-based game, only actions made by the current player are considered. Finally, meta-moves, such as resigning at a game of chess, have to be possible as well. Starting from this modelling of a game of chess, we model a Blockchain version of this game. Potential motivations to integrate Blockchain technology with the game of chess are to be able to easily bid on the outcome of a game and to secure the Elo rating [40] of players, a system ranking players based on their result history, on the Blockchain in an immutable and transparent manner. The only limitation of our proposed application is the possibility of cheating of one of the two players, by using of a chess-playing software. Cheating is one common occurrence in online chess gaming, and betting on the outcome of a game would be an incentive for players to do so. However, cheating detection in chess is an active research field [41].

In order to find the constraints of this DBA, we first need to have an ontology that defines the different concepts our game integrates. We present in Fig. 3 the ontology defining the concepts needed for our DBA. We use Protégé [42] to import the EthOn [28] ontology defined using OWL, and we extend it with additional concepts related to the chess game. Indeed, EthOn is one among the most complete existing ontologies related to Blockchain development. Building upon previous research facilitates collaboration, so we prefer to extend it instead of building a new ontology from scratch. Fig. 3 is a representation of part of the extended ontology, using rectangles to represent the concepts and arrows for properties. Firstly, the players are represented by the Blockchain account they use. This account is composed of a public and private key pair. Then, we define the different actions a player can make, which are then included into messages. Chess already has a standard notation, e.g. Be5 means "move a bishop to e5", so we do not have to redefine moves. A transaction is composed of four fields: an address "from", an address "to", a value, and arbitrary data. The messages describing user actions are included into the data field of transactions. As chess is a fully deterministic game, the game state is entirely defined by the move history. The game's logic is programmed using smart contracts. Indeed, we may not use the Blockchain to execute the logic, and decide to process each move off-chain, but we have to be able to verify the results in case of conflicts between the two players.

Along with this ontology, we also propose the workflow of the game using BPMN models. For the sake of simplicity, we focus on what happens when one of the players wants to make a move. The player first needs to transmit the information of his move to the other player. To do so, the application encapsulates the user's craft message describing the intended action. This message is then propagated to the Blockchain by including it in a new transaction. Once this transaction is sent, both players wait for the transaction to be included in a Block and for enough confirmations to have passed. The time needed for the transaction to be considered final can vary significantly, from one second to one hour, depending on the chosen Blockchain's characteristics and if a second layer scaling solution is used. During this time, no other action can be made by the players. Then, once the new Block has propagated, they each change their game states accordingly, and check whether the game follows an end game position, e.g. if one player checkmates the other. If it is not the case, the application awaits for the action of the second player.

This workflow is formalised in Fig. 4 showing a BPMN modelling of the game. This modelling is done using Camunda BPM [43]. Here, we use a high-level representation abstraction of the concepts of Blockchain and Nodes, as these are services from the viewpoint of our application. For example, the task "Generate block" hides the fact that the Block should be valid and follow the Blockchain's method of consensus. As the Ethereum Blockchain uses Proof of Work, this task requires the node to increment the nonce in the Block's header in order to find a valid Block. Once a valid Block is created, the node can transmit it to the rest of the network.

B. Services needed for the development of this DBA

Based on the ontology and the BPMN models presented above, we can deduce the services that are most suitable for an implementation of our target DBA. The ontology and BPMN are used to tell the expert each interaction he/she must focus on, by giving a comprehensive list of constraints. However, an expert still has to manually go through the existing solutions in order to see what the best one to use is. The automation of this process is future work.

On average, a game of chess ends at move 40. As a result, one constraint for our DBA is to be able to exchange 40 messages between the players at a reasonable cost. Based on the BPMN model presented, we can only use a Blockchain and send a transaction for each move. However, in this use case, the DBA's specifications should specify the maximum allowed latency between each move. As a result, services used to reduce latency below this threshold, such as state channels, sidechains or plasma, must be studied. We recommend using a second layer scaling solution called state channels [36]. Indeed, they are suitable for this application: the implementation of state channels between two participants is straightforward, and the cost is negligible. Furthermore, the smart contract handling the Blockchain logic for the game is made using Ethereum, as it is the most popular Blockchain with support for smart contracts, and is suitable for this case. Finally, no large data transfers are needed for this application. At most, we use IPFS solely to display the game's interface. This interface is an SPA hosted on IPFS, and accessible through a browser via either an IPFS gateway or a browser extension hosting an IPFS node. The choice of the IPFS provider is non-critical, and depends on the wanted accessibility and security for the application.

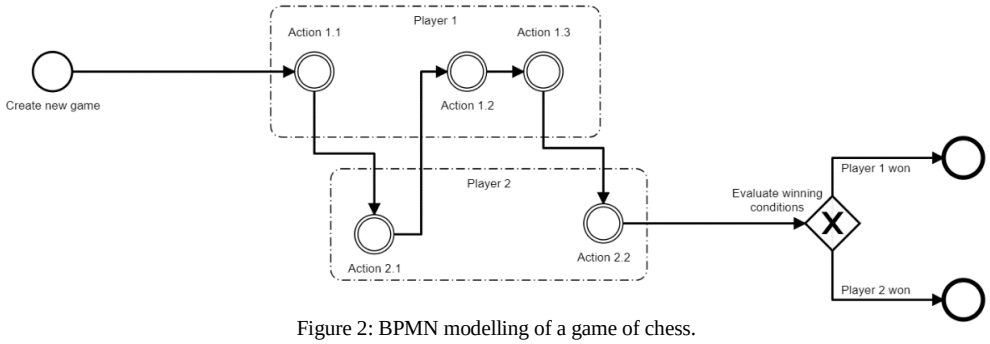

Figure 2: BPMN modelling of a game of chess.

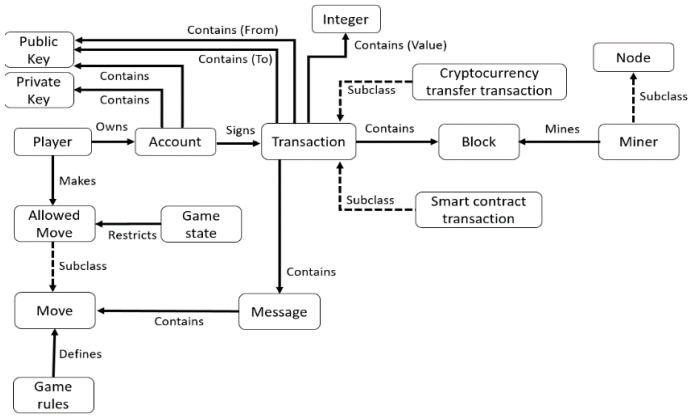

Figure 3: Partial view of an ontology defining the concepts needed for a Blockchain chess game.

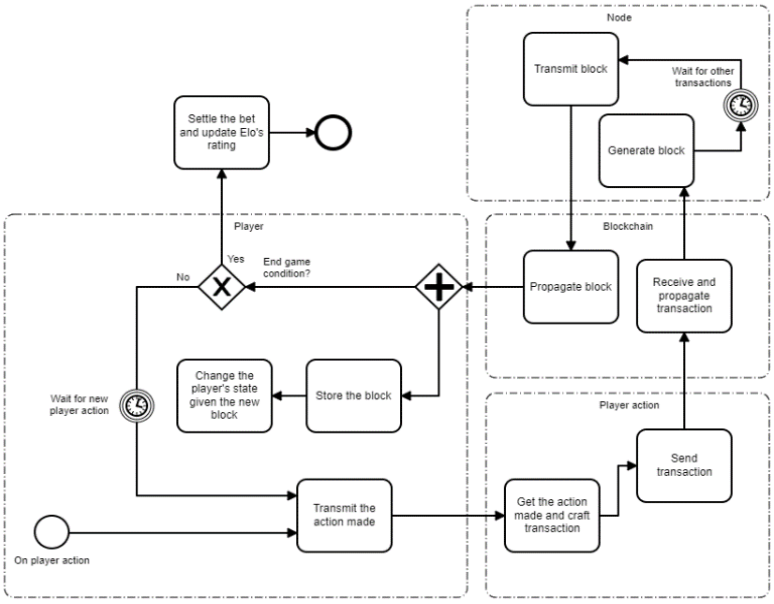

Figure 4: BPMN describing a Blockchain chess game.

V. CONCLUSIONS AND PROSPECTS

We propose a new approach for DBA architecture design, based both on business process modelling and ontologies. Our approach provides in the end a set of specific services for the implementation of a particular DBA.

Future work includes automating the process of choosing the most suitable service instances with recommendation tools based on multi-criteria algorithms. Finally, we intend to apply our methodology to other video games, such as real time multiplayer games.

REFERENCES

[1] Z. Zheng, S. Xie, H.-N. Dai, X. Chen and H. Wang, "Blockchain challenges and opportunities: a survey," *Int. J. Web and Grid Services,*, vol. 14, p. 352–375, 2018.

[2] PatAltimore, *Azure Blockchain Workbench architecture - Azure Blockchain.* [Online]. Available: https://docs.microsoft.com/enus/azure/blockchain/workbench/architecture

[3] R. Viswanathan, D. Dasgupta et S. R. Govindaswamy, «Blockchain Solution Reference Architecture (BSRA),» *IBM*

Journal of Research and Development, vol. 63, p. 1:1–1:12, 3 2019.

[4] H. M. Kim and M. Laskowski, "Towards an Ontology-Driven Blockchain Design for Supply Chain Provenance," *SSRN Electronic Journal*, 2016.

[5] S. Raval, Decentralized applications: harnessing Bitcoin's blockchain technology, " O'Reilly Media, Inc.", 2016.

[6] D. Treisman, «Defining and measuring decentralization: a global perspective,» 2002.

[7] G. Slepak et A. Petrova, «The DCS Theorem,» 2018.

[8] *Azarus game challenge network.* [Online]. Available: https://wp.azarus.io/

[9] I. Grigg, «EOS - An Introduction,» 7 2017. [Online]. Available: https://eos.io/documents/EOS An Introduction.pdf

[10] D. Larimer, *DPOS Consensus Algorithm - The Missing White Paper — Steemit.* [Online]. Available: https://steemit.com/dpos/@dantheman/dpos-consensus-algorithm-this-missing-white-paper

[11] A. Gervais, G. O. Karame, K. Wüst, V. Glykantzis, H. Ritzdorf et S. Capkun, «On the Security and Performance of Proof of Work Blockchains,» chez *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, New York, NY, USA, 2016.

[12] S. King and S. Nadal, "PPCoin: Peer-to-Peer Crypto-Currency with Proof-of-Stake," *Self-published paper*, p. 6, 2012.

[13] Maker, «The Dai Stablecoin System,» [Online]. Available: *https://makerdao.com/whitepaper/DaiDec17WP.pdf*, 2017.

[14] A. Biryukov, D. Khovratovich et S. Tikhomirov, «Findel: Secure Derivative Contracts for Ethereum,» chez *Financial Cryptography and Data Security*, vol. 10323, M. Brenner, K. Rohloff, J. Bonneau, A. Miller, P. Y. A. Ryan, V. Teague, A. Bracciali, M. Sala, F. Pintore et M. Jakobsson, Éds., Cham, Springer International Publishing, 2017, p. 453–467.

[15] C. Liu, K. K. Chai, E. T. Lau et Y. Chen, «Blockchain Based Energy Trading Model for Electric Vehicle Charging Schemes,» chez *Smart Grid and Innovative Frontiers in Telecommunications*, vol. 245, P. H. J. Chong, B. Seet, M. Chai et S. U. Rehman, Éds., Cham, Springer International Publishing, 2018, p. 64–72.

[16] *Gods Unchained TCG.* [Online]. Available: https://godsunchained.com

[17] Z. Duan, H. Mao, Z. Chen, X. Bai, K. Hu et J.-P. Talpin, «Formal Modeling and Verification of Blockchain System,» chez *Proceedings of the 10th International Conference on Computer Modeling and Simulation*, New York, NY, USA, 2018.

[18] T. Abdellatif et K. Brousmiche, «Formal Verification of Smart Contracts Based on Users and Blockchain Behaviors Models,» chez *2018 9th IFIP International Conference on New Technologies, Mobility and Security (NTMS)*, 2018.

[19] K. K. Ramachandran et B. Sikdar, «A Queuing Model for Evaluating the Transfer Latency of Peer-to-Peer Systems,» *IEEE Transactions on Parallel and Distributed Systems*, vol. 21, p. 367–378, 3 2010.

[20] CryptoKitties, *CryptoKitties | Collect and breed digital cats!*. [Online]. Available: https://www.cryptokitties.co

[21] *Cryptocurrency Exchange EtherDelta Hacked in DNS Hijacking Scheme*, 2017. [Online]. Available: https://www.ccn.com/cryptocurrency-exchange-etherdelta-hackedin-dns-hijacking-scheme/

[22] J. Benet, "IPFS - Content Addressed, Versioned, P2P File System," *arXiv:1407.3561 [cs]*, 7 2014.

[23] *Tug Of War, An Unstoppable Game.* [Online]. Available: http://tugofwar.io

[24] T. Cerny, M. J. Donahoo et M. Trnka, «Contextual Understanding of Microservice Architecture: Current and Future Directions,» *SIGAPP Appl. Comput. Rev.*, vol. 17, p. 29–45, 1 2018.

[25] W. Cai, Z. Wang, J. B. Ernst, Z. Hong, C. Feng et V. C. M. Leung, «Decentralized Applications: The Blockchain-Empowered Software System,» *IEEE Access*, vol. 6, p. 53019–53033, 2018.

[26] G. S. Ramachandran et B. Krishnamachari, «A Reference Architecture for Blockchain-based Peer-to-Peer IoT Applications,» *arXiv preprint arXiv:1905.10643*, 2019.

[27] H. Ugarte, A more pragmatic Web 3.0: Linked Blockchain Data, 2017.

[28] J. Pfeffer, «EthOn—introducing semantic Ethereum,» *blogpost, https://media.consensys.net/ethon-introducing-semantic-ethereum-15f1f0696986#.ttvx7c83i*, 2017.

[29] EverdreamSoft, *everdreamsoft/sandra.* [Online]. Available: https://github.com/everdreamsoft/sandra

[30] S. A. White, «Introduction to BPMN,» *Ibm Cooperation*, vol. 2, p. 0, 2004.

[31] J. Solís-Martínez, J. P. Espada, N. García-Menéndez, B. C. Pelayo G-Bustelo and J. M. Cueva Lovelle, "VGPM: Using business process modeling for videogame modeling and code generation in multiple platforms," *Computer Standards & Interfaces*, vol. 42, p. 42–52, 11 2015.

[32] C. Politowski, L. Fontoura, F. Petrillo et Y.-G. Guéhéneuc, «Are the Old Days Gone?: A Survey on Actual Software Engineering Processes in Video Game Industry,» chez *Proceedings of the 5th International Workshop on Games and Software Engineering*, New York, NY, USA, 2016.

[33] B. Magri, C. Matt, J. B. Nielsen and D. Tschudi, "Afgjort – A Semi-Synchronous Finality Layer for Blockchains," *Self-published paper*, p. 44, 2019.

[34] G. Decker et F. Puhlmann, «Extending BPMN for Modeling Complex Choreographies,» chez *On the Move to Meaningful Internet Systems 2007: CoopIS, DOA, ODBASE, GADA, and IS*, Berlin, 2007.

[35] M. Zamani, M. Movahedi et M. Raykova, «RapidChain: Scaling Blockchain via Full Sharding,» chez *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*, New York, NY, USA, 2018.

[36] J. Poon and T. Dryja, "The Bitcoin Lightning Network: Scalable Off-Chain Instant Payments," p. 59, 2016. [Online]. Available: https://www.bitcoinlightning.com/wpcontent/uploads/2018/03/lightning-network-paper.pdf

[37] L. Besançon, C. Ferreira Da Silva et P. Ghodous, «Towards Blockchain Interoperability: Improving Video Games Data Exchange,» chez . *IEEE International Conference on Blockchain and Cryptocurrency*, Seoul, 2019.

[38] R. Balzer et N. Goldman, «Principles of Good Software Specification and Their Implications for Specification Languages,» chez *Proceedings of the May 4-7, 1981, National Computer Conference*, New York, NY, USA, 1981.

[39] J. Zahnentferner, «Chimeric Ledgers: Translating and Unifying UTXO-based and Account-based Cryptocurrencies.,» *IACR Cryptology ePrint Archive*, vol. 2018, p. 262, 2018.

[40] A. E. Elo, The rating of chessplayers, past and present, Arco Pub., 1978.

[41] D. J. Barnes et J. Hernandez-Castro, «On the limits of engine analysis for cheating detection in chess,» *Computers & Security*, vol. 48, pp. 58-73, 2015.

[42] N. F. Noy, M. Crubézy, R. W. Fergerson, H. Knublauch, S. W. Tu, J. Vendetti et M. A. Musen, «Protégé-2000: an open-source ontology-development and knowledge-acquisition environment.,» chez *AMIA Annual Symposium Proceedings*, 2003.

[43] *Workflow and Decision Automation Platform.* [Online]. Available: https://camunda.com/