An Enterprise Architecture Approach to Semantic Blockchain Interoperability

Sebastião Sotto Mayor[1], Rafael Belchior[1,2], Miguel Correia[1] and André Vasconcelos[1]
[1]*INESC-ID, Instituto Superior Técnico, Universidade de Lisboa, Portugal*
[2]*Blockdaemon Ltd.*
sebastiao.sotto.mayor, rafael.belchior, miguel.p.correia, andre.vasconcelos@tecnico.ulisboa.pt

Keywords: Enterprise Architecture, Blockchain Interoperability, Ontology.

Abstract: Blockchain technology has revolutionized the way data is stored and accessed in a decentralized manner. However, the lack of interoperability between such systems is an ongoing challenge hindering their wider adoption. This document proposes a two-part solution composed of activities that aim to enhance semantic interoperability between homogeneous and heterogeneous blockchain systems. The first part are the design-time activities that consist of constructing an Archimate model, extracting its Resource Description Framework (RDF) ontology, and assessing its correctness utilizing a semantic reasoner. The second part are the runtime activities that involve leveraging the resulting ontology in a supply chain management application to validate transactions among participants in a network of systems. The evaluation results are promising, demonstrating that a shared ontology can support a transparent and accurate transaction validation approach. Thus, this work is a significant step in proving that distributed ledger technologies can benefit from enterprise architecture techniques to improve their interoperability.

1 INTRODUCTION

Distributed ledger technology (DLT) refers to networks of computers that store databases in a distributed way, distinguishing them from traditional databases confined to a single server. This decentralized structure enhances security by making data tampering more challenging for malicious entities. Blockchains store data in a chain of blocks and are a specific type of DLT. Each block contains multiple transaction records, and once appended to the chain, a block becomes immutable. This mechanism establishes a permanent and transparent ledger of all transactions conducted in a network. In addition to being used to record and track monetary transactions, blockchain technology can potentially support applications in several other domains, including supply chain management, voting systems, and identity verification (Peck, 2017).

Private or permissioned blockchains are distributed ledger systems where access to the network and validation of transactions is limited to an exclusive group of entities. These blockchains provide enterprises with several advantages in comparison with permissionless blockchains. Firstly, data confidentiality is enhanced since sensitive information is shared only among trusted participants. Secondly, they are highly scalable due to reduced network congestion and faster transaction processing since the number of participants is reduced compared to public blockchains. Additionally, private blockchains offer increased control and governance, enabling businesses to set rules and consensus mechanisms that align with their specific requirements. Furthermore, their usage enhances compliance, which makes them a valuable asset for various use cases in industries with high regulation (Underwood, 2016).

Interoperability, which refers to the ability of different systems and components to work together efficiently, is a quality attribute that has four layers: the technical (further subdivided into functional and structural), the semantic (including interpreting information format and its meaning), the organizational, and the legal (Kalogirou and Charalabidis, 2019). Regarding specifically the semantic layer of interoperability, on the one hand information format pertains to the structure and representation of data exchanged between systems. This includes data types, encoding methods, and communication protocols. On the other hand, understanding the meaning embedded in the information involves interpreting the data context, semantics, and relationships, ensuring that the

exchanged information is correctly understood by the systems consuming it.

Over the last few years, there have been many attempts to introduce a solution that will mitigate the interoperability issues of blockchain systems (Belchior et al., 2021; Belchior et al., 2023b). However, few solutions focus on the semantic layer. While each layer has its significance, the semantic layer will play a key role in increasing interchain communication efficiency (Hardjono et al., 2019). By building systems on a common ground of understanding, actions conducted among systems with a high volume of cross-chain transactions become faster, as there is less friction to translate concepts from one blockchain to the other.

The objective of this work is to explore enterprise architecture (EA) mechanisms to produce an ontology (Antunes et al., 2013) that individual systems can leverage to build plugins, which are small and flexible components executing their business logic on top of a base architecture that includes the necessary elements to achieve interoperability. EA offers standardizable methods in the form of models that can facilitate the intercommunication of distributed ledger technology systems in the semantic layer by reducing ambiguity in valuable processes such as interchain transactions.

A proof-of-concept (POC) presents the practical aspects of this approach. Its scope is a network of permissioned blockchains owned by organizations with interoperation needs with fellow members of a supply chain consortium. The decision to restrict the POC to private blockchains comes from the fact that systems involving public blockchains operate with some inherently different concerns and requirements as mentioned above. Therefore, designing an EA model targeting both would complicate the solution prematurely.

The results extracted from the conducted experiments are encouraging, showing that the integrity of the transaction validation process is enhanced when following the proposed approach.

In summary, this paper introduces an EA that supports semantic interoperability between blockchain systems and implements a POC that will leverage the ontology extracted from its model. By proving the applicability and effectiveness of the proposed solution, the aim is to show that EA techniques can be valuable for solving interoperability issues in blockchain systems.

This paper is structured as follows. Section 2 gives an overview of the architecture and use case that support the POC, followed by Section 3, which describes the proposed solution in detail. Next, Section 4 provides the evaluation strategy and its outcome. Lastly, in Section 5, the authors discuss the conclusions drawn from this study and outline future work.

2 PRELIMINARIES

This section presents important knowledge concerning the proposed solution. Specifically, it provides an in-depth analysis of Hyperledger Cacti (Montgomery et al., 2020), as it forms the basis of the architecture that supports business logic plugins (BLPs) used for the interoperation of supply chain-related entities. Additionally, it describes the use case that will demonstrate the capabilities of the proposed solution along the remainder of the paper.

2.1 Hyperledger Cacti

Blockchain gateways are intermediary systems that facilitate interoperability between different blockchain networks. They act as bridges, enabling data transfer across disparate blockchain platforms, allowing them to exchange information and assets. One example of a middleware framework that adopts the concept of blockchain gateways is Hyperledger Cacti. Built on the Hermes model (Belchior et al., 2022), Cacti is an open-source solution that provides a fault-tolerant middleware layer supporting blockchain interoperability.

By implementing the protocols and mechanisms introduced by Hermes, Cacti enables secure and reliable communication between diverse blockchain networks, opening up opportunities for cross-chain transactions and the execution of smart contracts, which are self-executing contracts with the terms of the agreement written in a programming language like Solidity (Dannen, 2017). These contracts automatically execute and enforce themselves when predetermined conditions are met, without the need for intermediaries or third parties.

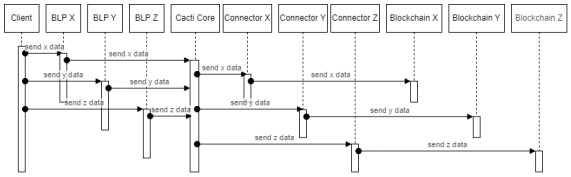

Figure 1: UML sequence diagram of an architecture supported by Hyperledger Cacti

Figure 1 depicts the high-level interactions of components utilizing Cacti. The client communicates with multiple BLPs via Representational State Transfer (REST) APIs. The BLPs then send the data to

Cacti Core, which serves as the central component responsible for coordinating the exchange of data between the plugins and the connectors via connector APIs. Lastly, each connector is responsible for interfacing with the respective blockchain.

2.2 Supply Chain Use Case

The use case chosen for the POC focuses on the supply chain domain. This choice comes from the fact that this business sector inherently utilizes ambiguous terms to describe processes, making it adequate for demonstrating the value of the proposed solution.

The use case consists of three sequential data transfers among three entities, where each entity leverages a different type of blockchain for business-specific reasons irrelevant to the study. A data transfer is the process of copying information from one DLT to another, and it involves specific data (e.g., manufacturer information) in most cases. This operation can involve multiple steps, including an optional intermediate processing step that may manipulate or analyze the data before sending it to the target chain. Data transfers are vital for ensuring the interoperability and exchange of valuable information across blockchains (Belchior et al., 2023a).

The data model comprises a business role, function, item, and process. That is, a *Supplier* may *Supply* a *Bamboo Harvest* to a *Manufacturer* in the context of a *Product Sale* and so on. There is an obvious concern for preserving privacy through the entire interoperation process among the three entities, as in most cases, the transferred data includes sensitive business information.

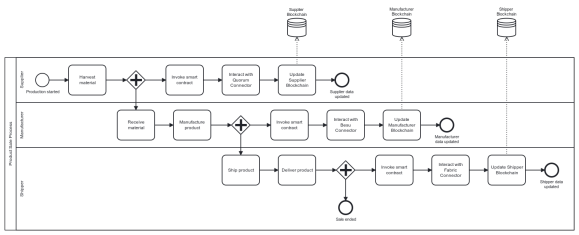

Figure 2: Supply chain use case BPMN diagram

Figure 2 depicts a sample flow of transactions taking place among the three entities of the use case as a Business Process Model and Notation (BPMN) diagram (White, 2004). To achieve interoperability between the three entities, Cacti is used. When an event occurs in one system, a smart contract is invoked. The Cacti Core then interacts with the appropriate Connector (Fabric, Besu, or Quorum) to update the respective blockchain.

3 ONTOLOGICALLY-GUIDED INTEROPERABILITY

This section presents a solution for achieving interoperability between two or more blockchain systems by leveraging a common ontology that derives from an EA model. The approach is divided into design-time and runtime activities of the interoperation process to ensure standardization. The former are the activities that take place before the interoperation process has begun. The latter are the activities that take place during the interoperation process.

3.1 Design-Time Activities

At design time, a model is created using Archimate, a modelling language for EA (Josey et al., 2016). This model is the foundation for producing a shared vocabulary as an RDF/XML ontology which represents the structural and conceptual aspects of the Archimate model in a machine-readable format (Decker et al., 2000). Lastly, the HermiT reasoner is used to assess the correctness of the extracted ontology (Bansal and Chawla, 2014).

3.1.1 Enterprise Architecture Model Design

Ideally, this process is conducted by consortium stakeholders to ensure alignment among participating entities. As they possess rich insights about the meaning of terms and the specific format of messages of each business area, close inter-collaboration determines the scope and contents of the EA model. Therefore, this should be an iterative process, and the model may change significantly as the knowledge grows. Furthermore, the European Interoperability Reference Architecture (EIRA) (Commission, 2015) should guide the design to ensure regulatory compliance.

Figure 3 depicts the three layers of the supply chain-specific EA created for the use case that was defined in Subsection 2.2. The upper layer portrays the Business layer with the relations between the key business entities. Those are the business roles, functions, processes, and items that will serve as the main study of this paper. The middle layer represents the Application layer with the software applications that support the business layer. It includes components such as application services and blockchain interfaces. Finally, the bottom layer illustrates the Technology layer with the technological infrastructure that supports the application layer. Its elements are the infrastructure and networks that enable blockchain transactions.

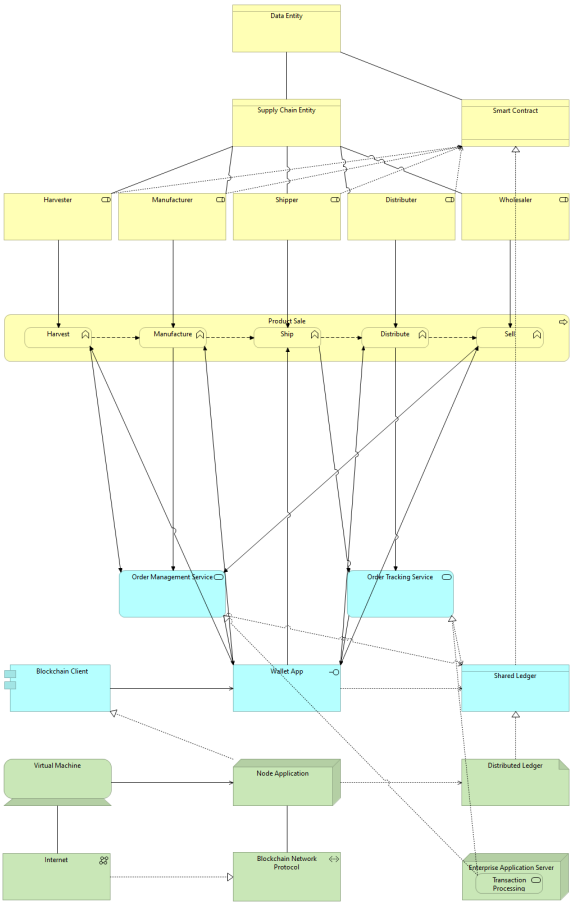

Figure 3: Supply chain blockchain interoperability EA model in Archimate

It is worth mentioning that only the upper layer supports the runtime activities described in Subsection 3.2 and that the middle and lower layers were included in the EA to illustrate the relations among the components of the supply chain ecosystem.

3.1.2 Ontology Extraction and Assessment

After designing the Archimate model, the next step is to extract its model exchange file in XML and convert it into an RDF ontology. A third-party tool designed specifically for converting Archimate models into RDF format handles this.

An RDF ontology is favored for managing data from multiple sources because unlike alternative ontology technologies, it accommodates various data types, ensuring seamless integration among diverse systems. Moreover, its semantic richness enables it to capture nuanced meanings and contextual relationships within the information.

Once the RDF ontology is ready, the assessment process using HermiT can proceed. HermiT is a reasoner that can analyze ontologies and infer additional knowledge based on the logical rules defined in the ontology. This process helps identify redundancies and missing information within the ontology.

HermiT can also check for the satisfiability and consistency of the ontology. Satisfiability determines if all the defined classes of the ontology have at least one instance, while consistency checks if there are no logical contradictions within the ontology. These checks are essential to ensure the ontology accurately represents the intended knowledge.

By using HermiT to assess the RDF ontology derived from the Archimate model, one gains valuable insights into the quality and correctness of the ontology. This analysis can help uncover potential issues or improvements in the design of the initial model, ensuring that the resulting ontology aligns with the intended architectural knowledge.

Listing 1, displays the format of a sample business entity in the ontology.

```
<NamedIndividual
      rdf:about="urn:uuid:id-6d7fc1370fcb48959daae518ea90b265">
   <rdf:type
         rdf:resource="http://bp4mc2.org/def/archimate#BusinessRole"/>
   <archimate:association
         rdf:resource="urn:uuid:id-67ccb819d7284462a8b9045f23e437d0"/>
   <archimate:triggering
         rdf:resource="urn:uuid:id-00f2f859356543cb8c0b8212e2731e1c"/>
   <archimate:writeAccess
         rdf:resource="urn:uuid:id-a5e7d160476c449ca342489c9b3f3d9f"/>
   <rdfs:label xml:lang="en">Manufacturer</rdfs:label>
</NamedIndividual>
```

Listing 1: Sample *Manufacturer* business role entity

3.2 Runtime Activities

At runtime, the integration process unfolds through a series of steps. First, the transaction context is defined, which consists of stating the target business role and the business process in which the transaction occurs. Afterwards, the source system sends the data to the shared Cacti BLP middleware. There, the data is mapped and converted to conform to the ontology. Finally, the BLP validates the data against the predefined constraints and rules. If the data is deemed valid, it is delivered to the target system and subsequently updated in the target blockchain.

3.2.1 Entity Mapping and Conversion

Mapping refers to retrieving the name of a relevant entity from the ontology. It can be either the same entity or a semantically similar one. Conversion refers to building the corresponding ontology role object based on the source role object by extracting all the available data. Both mapping and conversion pro-

cesses involve querying the ontology in SPARQL by utilizing an open-source library called rdflib.js (Taelman et al., 2018).

The first step is to create an rdflib.js store object that holds the ontology data. This store acts as a container for RDF triples, where each triple consists of a subject, predicate, and object. The store then reads the ontology file and populates itself with the existing triples in the ontology.

SPARQL, a standard query language for RDF data, is used to query the store. rdflib.js provides a convenient API to execute queries on the store. A SPARQL query can specify conditions to filter entities based on specific criteria, such as their properties or relationships with other entities. For example, one can query for all entities of a specific type or entities that possess a particular property value.

3.2.2 Transaction Validation

The transaction validation process consists of two steps. The first is data validation, and the second is state validation. Successful completion of both steps ensures a valid transaction. If either step is unsuccessful, the transaction cannot occur, and the remaining process should abort.

Algorithm 1 Data Validation Algorithm

Input: ontologyRoleObj: Object
Output: isValid: Boolean

Function `validateData` (*ontologyRoleObj*)
    fieldsToCompare ← getFieldsContainingWord(ontologyRoleObj, "Format")
    for *i* ← 0 **to** *length of fieldsToCompare* **do**
        formatFieldName ← fieldsToCompare[i] fieldName ← remove "Format" from formatFieldName
        if *fieldName not in ontologyRoleObj* **then**
            continue
        end
        fieldValue ← ontologyRoleObj[fieldName] formatValue ← ontologyRoleObj[formatFieldName]
        if *formatValue is not a regular expression* **then**
            return false
        end
        if *fieldValue does not match the format specified in formatFieldName* **then**
            return false
        end
    end
    return true
end

Data validation refers to checking whether the properties of the object are valid. Specifically, it ensures that the values are within the expected boundaries stipulated in the EA model. This step is depicted in Algorithm 1. This algorithm validates data in the ontology role input object by checking specific fields containing the word *Format*. For each of these fields, it ensures a corresponding non-format field exists. If not, it moves to the next field. The code validates the format field as a regular expression; if invalid, it returns false. It then checks if the non-format field matches the specified pattern. If not, it also returns false. If all checks pass, it returns true.

A concrete example of the above would be the sample transaction data in Listing 2:

```
{
    nameFormat: "^[a-zA-Z0-9\\-,. ]+$",
    name: "ABC suppliers"
}
```

Listing 2: Sample format and non-format fields

Algorithm 2 State Validation Algorithm

Input: sourceRole: String, targetRole: String, process: String
Output: isValid: Boolean

Function `validateState` (*sourceRole, targetRole, process*) **:**
    if *sourceRole is empty* **then**
        return false
    end
    processObj ← RDFQuerier.fetchDetailsByLabel(process) **if** *processObj is null* **then**
        return false
    end
    foreach *funcUri in processObj.get('archimate#composition')* **do**
        funcName ← RDFQuerier.fetchLabelByUri(funcUri) func ← find function in this.businessFunctions where func.entity equals funcName
        if *func.triggeredBy equals sourceRole* **then**
            sourceFunction ← func
        end
        if *func.triggeredBy equals targetRole* **then**
            targetFunction ← func
        end
        add func to funcs
    end
    return (sourceFunction ≠ null) ∧ ((targetFunction ≠ null ∧ areAdjacent(sourceFunction, targetFunction, funcs)) ∨ ¬targetRole)
end

State validation refers to assuring that the transaction is legal. In other words, a transition from or to specific business functions can occur in the current business process according to the EA model. This step is depicted in Algorithm 2. This algorithm performs a validation check on the input parameters *sourceRole*, *targetRole*, and *process*. It first ensures that *sourceRole* is not empty; if it is, the method returns false. Then, it attempts to fetch a list of functions related to the specified *process* using an RDF query and returns false if no functions are found.

The method then iterates through the list of functions, checking if any of them are triggered by the given *sourceRole* or *targetRole*. The method returns true if *sourceFunction* is not null and either *targetFunction* is not null and the *areAdjacent* method confirms the adjacency of *sourceFunction* and *targetFunction*, or if *targetRole* is empty.

4 EVALUATION

It is essential to obtain a comprehensive understanding of the advantages and limitations of a novel approach. Conducting both accuracy and performance analyses allows for a rigorous assessment of its capabilities across various dimensions.

Accuracy analysis scrutinizes the reliability of the system irrespective of its operational load. This examination serves the purpose of discerning the inherent capabilities of the system under controlled conditions. It is also pivotal in ensuring that the system effectively fulfills its intended objectives and operates correctly, laying the foundation for a thorough evaluation.

Conversely, performance analysis explores the system's behavior under diverse load conditions, simulating real-world scenarios of varying demand and stress levels. This data enables the assessment of scalability, resilience, and the system's ability to perform consistently in a production environment.

Integrating accuracy and performance analyses is crucial for a holistic evaluation. This method minimizes the risk of oversight and mitigates potential biases. Furthermore, it supports the principle of continuous improvement. As the system evolves, periodic reevaluation through these lenses can assist in addressing emerging challenges.

4.1 Experimental Setup

The experimental infrastructure consisted of three test ledgers (one for each supply chain entity) interoperating with Cacti. The three test ledgers were Besu (Fan et al., 2022), Quorum (Hounsom, 2018), and Fabric (Androulaki et al., 2018). These ledgers simulated different blockchain networks and enabled the deployment and interaction with smart contracts. Each component ran on a separate application container on a single machine equipped with an AMD Ryzen 7 3750H processor and 16GB of RAM.

The deployment of smart contracts was a critical aspect of the experimental setup, as each of them handled different supply chain aspects. The *BambooHarvestRepository.sol* and *BookshelfRepository.sol* contracts were written in Solidity and the former managed bamboo harvest records, while the latter handled bookshelf records. The *Shipment.ts* script written in Go represented a smart contract for managing shipments. The bamboo harvest and bookshelf smart contracts were deployed in the Besu and Quorum blockchains respectively while the shipment smart contract was deployed on the Fabric ledger.

The setup process also involved creating test accounts, generating contract addresses, and storing contract artifacts in the keychain plugin to manage cryptographic keys securely. It also included specifying the target organizations, channel, and policy that determines which members of each organization can approve a transaction involving a specific smart contract.

The purpose of the above setup was to provide a self-contained and local environment that allowed for assessing the effectiveness of the proposed solution by evaluating the POC that puts it into practice. Thus, it provided the necessary functionality for the experiments without requiring a complex and time-consuming process.

4.2 Accuracy Analysis

The chosen metric for the accuracy analysis was the error rate of the POC when validating transactions (Tx) among blockchain systems. Therefore, the approach followed was to conduct experiments with valid and invalid transactions to ascertain if the solution correctly identified each case.

Each validation process scrutinizes two main elements. Those are the transaction context and the object that initiates the transaction. If either of them is invalid, the transaction should not complete.

Specifically, a context is valid when its target role and business process entities are present in the ontology. Similarly, an object is in a legal state when its source role entity exists in the ontology and its fields conform to the specified format.

In the first experiment, both the context and the object were valid. Here the transaction was deemed successful as expected because all steps passed.

The second experiment had an invalid context and valid object. This time, despite the mapping step executing successfully, the validation step failed because of its invalid context.

The third experiment involved a context that was valid and an invalid object. Here, the mapping step was unsuccessful, so the validation failed as it did not even get a chance to execute.

In the fourth experiment, both the context and the object were invalid. Again, the mapping failed, so the

transaction validation was deemed unsuccessful.

The experiment results are summarized in Table 1.

Table 1: Experiment results

#	Context	Object	Valid Tx	Expected
1	Valid	Valid	✓	Yes
2	Invalid	Valid	×	Yes
3	Valid	Invalid	×	Yes
4	Invalid	Invalid	×	Yes

4.3 Performance Analysis

The metric used for the performance analysis was the processing time to complete a transaction among two separate blockchain systems in the POC.

Given that the proposed solution is supported by the infrastructure of Hyperledger Cacti, it is worth noting that the evaluation of the performance of the former is framed to that of the latter.

Experiments were conducted for three distinct load profiles characterized as ”low,” ”normal,” and ”high,” corresponding to a rate of 10, 50, and 100 simultaneous transactions. These three profiles represent typical loads in enterprise ecosystems with a scope similar to the one presented in this paper.

To extract objective and unbiased results, each transaction load profile execution was repeated 100 times and their mean time was calculated. While performing the first transactions, the systems may not have reached their processing potential, so the first iterations can be considered a warm-up phase. Furthermore, worker threads were utilized to permit the parallel execution of the transactions.

Figure 4 depicts the impact on processing time as the rate of simultaneous transactions increases before and after integrating the proposed solution. One may interpret the graph as the processing overhead imposed on the participating systems when leveraging the approach presented in this work.

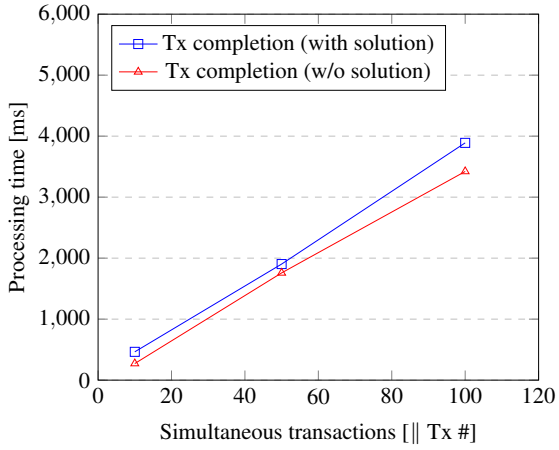

Figure 4: Overhead of proposed solution

4.4 Result Discussion

The experiments show that the POC leveraging the proposed solution identifies valid and invalid transactions among systems with 100% accuracy. The benefits of this outcome are immense, as the interoperation process becomes transparent and efficient.

The increase in the processing overhead while utilizing the proposed solution indicates possible scalability issues. Nevertheless, the overhead is considered negligible for enterprise standards until a certain point. Thus, it is suitable for private ecosystem usage where simultaneous transactions typically do not exceed this threshold, and data quality is of foremost concern.

In other words, it is acceptable to sacrifice system performance for reliability. As a result, the capabilities of the individual blockchain systems get enhanced by reducing duplication of effort, improving data integrity and consistency, and enabling new decentralized applications and services to be built on top of them securely.

Another observation is the low cost and ease of extensibility of the solution. While the design and consequent change management of modifying the EA model involve resources, the possibility of automation of the remaining processes implies that spending does not increase for those.

The applicability of the proposed solution is ample. Every network that involves complex transactions among participating systems would gain from this solution. Thus, this approach could potentially impact many industries that actively leverage blockchains and, perhaps, encourage others to migrate their current infrastructure to DLT.

5 CONCLUSIONS

There are a plethora of solutions related to achieving interoperability in blockchain systems. Nevertheless, most of them do not focus on the semantic layer of interoperability. This paper explored this gap by implementing a solution that enhances interoperation among systems owned by members of a supply chain consortium, as those usually entail operations among different business domains, demonstrating the potential of this mechanism. The novel approach leverages EA modeling to provide a common ground for systems to achieve interoperability in the semantic layer. It consists of an ontologically guided business logic plugin that validates cross-chain transactions. The outcome of the evaluation is satisfactory, proving the added value of this approach. Although the POC is

limited to a network of permissioned ledgers owned by a business consortium, the paradigm can be extended for a set of blockchain systems of indefinite size. Thus, this paper supports that EA techniques can aid in solving open issues in blockchain interoperability and, in this way, encourage further research in this area.

This work needs to be generalized for ease of extensibility to other use cases. For instance, applying this mechanism to enhance public blockchain interoperability in fields other than that of supply chain management would be noteworthy. Another interesting future research path is to extend the Hermes fault-tolerant middleware that was described in the preliminary section of this document. The goal would be to increase the robustness of this tool by leveraging the transaction validation capabilities offered by the proposed solution.

ACKNOWLEDGEMENTS

We thank the reviewers of this paper for their very insightful and constructive feedback. We put forward our gratitude to the Hyperledger Cacti community, which idealized the use case we explore in this paper.

This work was supported by national funds through FCT, Fundação para a Ciência e a Tecnologia, under project DOI:10.54499/UIDB/50021/2020 and in the scope of the project nr. 51 "BLOCKCHAIN.PT - Agenda Descentralizar Portugal com Blockchain" refª C632734434-00467077, financed by European Funds, namely "Recovery and Resilience Plan - Component 5: Agendas Mobilizadoras para a Inovação Empresarial", included in the NextGenerationEU funding program.

REFERENCES

Androulaki, E., Barger, A., Bortnikov, V., Cachin, C., Christidis, K., De Caro, A., Enyeart, D., Ferris, C., Laventman, G., Manevich, Y., et al. (2018). Hyperledger fabric: a distributed operating system for permissioned blockchains. In *Proceedings of the thirteenth EuroSys conference*, pages 1–15.

Antunes, G., Caetano, A., Bakhshandeh, M., Mayer, R., and Borbinha, J. (2013). Using ontologies for enterprise architecture model alignment. In *Proceedings of the 4th Workshop on Business and IT Alignment (BITA 2013). Poznan, Poland*.

Bansal, R. and Chawla, S. (2014). An approach for semantic information retrieval from ontology in computer science domain. *International Journal of Engineering and Advanced Technology (IJEAT)*, 4(2):58–65.

Belchior, R., Riley, L., Hardjono, T., Vasconcelos, A., and Correia, M. (2023a). Do you need a distributed ledger technology interoperability solution? *Distributed Ledger Technologies: Research and Practice*, 2(1):1–37.

Belchior, R., Süßenguth, J., Feng, Q., Hardjono, T., Vasconcelos, A., and Correia, M. (2023b). A brief history of blockchain interoperability.

Belchior, R., Vasconcelos, A., Correia, M., and Hardjono, T. (2022). Hermes: Fault-tolerant middleware for blockchain interoperability. *Future Generation Computer Systems*, 129:236–251.

Belchior, R., Vasconcelos, A., Guerreiro, S., and Correia, M. (2021). A survey on blockchain interoperability: Past, present, and future trends. *ACM Computing Surveys (CSUR)*, 54(8):1–41.

Commission, E. (2015). Eira support series - how eira supports interoperability v1.0.0.

Dannen, C. (2017). *Introducing Ethereum and solidity*, volume 1. Springer.

Decker, S., Melnik, S., Van Harmelen, F., Fensel, D., Klein, M., Broekstra, J., Erdmann, M., and Horrocks, I. (2000). The semantic web: The roles of xml and rdf. *IEEE Internet computing*, 4(5):63–73.

Fan, C., Lin, C., Khazaei, H., and Musilek, P. (2022). Performance analysis of hyperledger besu in private blockchain. In *2022 IEEE International Conference on Decentralized Applications and Infrastructures (DAPPS)*, pages 64–73. IEEE.

Hardjono, T., Lipton, A., and Pentland, A. (2019). Toward an interoperability architecture for blockchain autonomous systems. *IEEE Transactions on Engineering Management*, 67(4):1298–1309.

Hounsom, C. (2018). Quorum whitepaper v0.2.

Josey, A., Lankhorst, M., Band, I., Jonkers, H., and Quartel, D. (2016). An introduction to the archimate® 3.0 specification. *White Paper from The Open Group*.

Kalogirou, V. and Charalabidis, Y. (2019). The european union landscape on interoperability standardisation: status of european and national interoperability frameworks. In *Enterprise Interoperability VIII: Smart Services and Business Impact of Enterprise Interoperability*, pages 359–368. Springer.

Montgomery, H., Borne-Pons, H., Hamilton, J., Bowman, M., Somogyvari, P., Fujimoto, S., Takeuchi, T., Kuhrt, T., and Belchior, R. (2020). Hyperledger cactus whitepaper.

Peck, M. E. (2017). Blockchains: How they work and why they'll change the world. *IEEE Spectrum*, 54(10):26–35.

Taelman, R., Van Herwegen, J., Vander Sande, M., and Verborgh, R. (2018). Comunica: a modular sparql query engine for the web. In *The Semantic Web–ISWC 2018: 17th International Semantic Web Conference, Monterey, CA, USA, October 8–12, 2018, Proceedings, Part II 17*, pages 239–255. Springer.

Underwood, S. (2016). Blockchain beyond Bitcoin. *Communications of the ACM*, 59(11):15–17.

White, S. A. (2004). Introduction to bpmn. *Ibm Cooperation*, 2(0):0.