Velluscinum: A Middleware for Using Digital Assets in Multi-Agent Systems

Nilson M. Lazarin[1,2], Igor M. Coelho[1], Carlos E. Pantoja[1,2], and José Viterbo[1]

[1] Institute of Computing - Fluminense Federal University (UFF), Niterói - RJ, Brazil
[2] Federal Center for Technological Education Celso Suckow da Fonseca (Cefet/RJ), Rio de Janeiro - RJ, Brazil
{nilson.lazarin,carlos.pantoja}@cefet-rj.br, {imcoelho,viterbo}@ic.uff.br

Abstract. Distributed Ledger Technologies (DLT) characteristics can contribute to several domains, such as Multi-agent Systems (MAS), facilitating the agreement between agents, managing trust relationships, and distributed scenarios. Some contributions to this integration are in the theoretical stage, and the few existing practical contributions have limitations and low performance. This work presents a MAS approach that can use digital assets as a factor of agreement in the relationship between cognitive agents using the Belief-Desire-Intention model. To validate the proposed methodology, we present the middleware Velluscinum that offers new internal actions to agents. The middleware was tested by adapting the Building-a-House classic example to cryptocurrency and agreements mediated by a distributed ledger.

Keywords: Middleware · Multi-agents · Digital Ledger Technology.

1 Introduction

Multi-agent Systems (MAS) are systems composed of multiple agents, which can be cognitive, through the use of the Belief-Desire-Intention (BDI) [4] model that enables the programming of mental attitudes like beliefs, desires, and intentions. These agents are called cognitive since they have a reasoning cycle capable of analyzing the information perceived in the environment in which they are inserted, the knowledge acquired through communication with other agents, and making self-conclusions. These agents can interact or compete to resolve a task; in this relationship, conflicts and uncertainty can occur, so it is essential to provide mechanisms for all agents to cooperate by guaranteeing reliability in the acquired information and used resources [23, 25].

Distributed Ledger Technologies (DLT) technologies have applications far beyond the financial sector, and their characteristics of decentralization, security, trust, and low cost of operation have a great capacity to contribute in various domains [9]. They are classified as permissionless or permissioned: in the first case, the access is unrestricted, the members can join and leave the network at any time, and finally, each node has a read-only copy of the data; the second case provides an additional access control layer, allowing some specific operations by authorized participants [18].

Adopting DLT technologies in MAS can facilitate the agreement between agents, taking what is registered in the Ledger as accurate, and also can be helpful to manage trust relationships, open MAS, and distributed scenarios. This integration contains a great set of open challenges with great potential [6]. Whether in facilitating the execution of semi-autonomous interorganizational business processes [11] or even allowing intelligent agents to generate economic value for their owner [14], for example.

Many contributions that proposed the fusion of MAS and DLT are still in the theoretical stage [6]. When performing a mapping review, we found only one paper [16] implementing DLT in MAS using BDI agent. However, this implementation is in the environmental dimension of the MAS, making it necessary to create an institution in the organizational dimension. In this implementation, the agents can access only one wallet in the DLT, limiting the competitiveness or autonomy of agents. In addition, the DLT platform used has high latency and low performance.

Thus, this paper presents a middleware for using digital assets in the relationships between cognitive agents to represent the transfer of funds, registration of ownership of artifacts, declaration of promises or agreements, and dissemination of knowledge. The middleware comprises several new internal actions that operate in the agents' dimension, allowing them to manipulate assets and wallets directly in the DLT. So, the agents can create and transfer divisible and indivisible assets and manage digital wallets on a permissioned DLT. In addition, as a proof of concept, a case study integrating MAS and DLT is presented. They adapted the well-known Build-a-House example [1] for using digital assets in a BigchainDB Network. The contribution of this work is a middleware to be integrated in the JaCaMo [2] distribution, a well-known platform for agent-oriented development.

This work is organized as follows: a theoretical basis of DLT is presented in Section 2; an analysis of related works is presented in Section 3; in Section 4 is presents a proposal for the use of digital assets in the relationship between intelligent agents; the case study of the integration of the well-known examples with DLT is presented in Section 5; finally, conclusions and future work are presented in Section 6.

2 Theoretical foundation

A DLT can be considered an append-only decentralized database because it provides a storage mechanism. However, compared with a traditional database, its performance is much lower because it has a low download rate and a high latency [13]. On the other hand, there are several models of permissioned DLTs, but they do not have a significant difference in performance compared to a permissionless [8, 22].

New approaches based on distributed databases have been used to improve the performance of permissioned DLT. In this case, the properties of distributed databases are combined with blockchain networks, are thus security-oriented, and adopt transaction-based replication [21, 17, 10]. When considering the number of transactions that a DLT can successfully execute per second (*Transactions Per Second – TPS*), some benchmark works carried out demonstrate that: DLT permissionless (e.g., Bitcoin or Ethereum) ranges between 3.45 and 4.69 TPS [24, 15]; DLT permissioned (e.g., HyperLedger Fabric) vary between 4.28 and 10.51 TPS [7, 15]; DLT permissioned distribution database-based (e.g., BigchainDB) varies between 50.60 and 175 TPS [7, 10].

The performance of DLT can be a limiting factor for their adoption in MAS since the delay in including a new block in the network can affect the behavior of intelligent systems. In addition, the cost of carrying out a transaction on some permissionless DLT can be another restrictive factor for wide use. These issues led to the choice of using BigchainDB, as it is a high-performance permissioned network. Rather than trying to improve the performance of DLT, BigChainDB adds the characteristics of a blockchain to a distributed database. In this way, it has unified the characteristics of low latency, high transfer rate, high storage capacity, and a query language of a distributed database, with the characteristics of decentralization, Byzantine fault tolerance, immutability, and creation or exchange of digital assets [13].

BigchainDB structures the data as an owner-controlled asset and only allows two types of transactions that are made up of the following fields [13]: *ASSET* is immutable information that represents a register in the DLT; *METADATA* is additional information that can be updated with each new transaction; *INPUT* specifies which key an asset previously belonged to and provides proof that the conditions required to transfer ownership of that asset have been met; *OUTPUT* specifies the conditions that need to be met to change ownership of a specific asset; *TRANSACTION-ID* is a digest that identifies the transaction. It is computed considering all transaction fields.

There is no previous owner in a *CREATE* transaction, so the *INPUT* field specifies the key registering the asset. Furthermore, in the *OUTPUT* field of this transaction, a positive number called *AMOUNT* is defined. If $AMOUNT = 1$, the asset is indivisible, thus representing a non-fungible token. If $AMOUNT > 1$, this asset is divisible, thus representing a token and how many instances there are. In a *TRANSFER* transaction, the INPUT contains proof that the user can transfer or update that asset. In practical terms, this means that a user states which asset is to be transferred with the *INPUT* field and demonstrates that it is authorized to transfer this asset [13].

3 Related Works

A mapping review was conducted by looking for related works integrating BDI agents with DLT. This research was made in three phases. The first phase used the following search string *((“DLT” OR “Distributed Ledger Technologies” OR*

"Blockchain") AND ("BDI" OR "belief-desire-intention") AND ("multi-agent oriented programming" OR "Jason" OR "Jacamo")) in GoogleScholar found 97 works. In the second phase, a refinement was carried out by reading the paper's titles and abstracts. Twenty works were classified involving DLT and MAS. In the third stage, the complete reading of the papers. Below are the only three papers that involved DLT and MAS practical applications.

Calvaresi et al. [6] presents an integration between the MAS and the Hyperledger Fabric. Via a smart contract, the reputation management of agents of a MAS is carried out. In the proposed model, prior registration is required to interact with other agents in the system or operate the ledger, making the use of a third membership service mandatory. Furthermore, an agent cannot create an asset in the ledger, but only execute a pre-existing contract. Finally, the proposed system does not use BDI agents.

Papi et al. [16] presents an integration model between JaCaMo and Ethereum. Creating a centralized entity and standards allows BDI agents to request the execution of a smart contract on the blockchain. A proof of concept is presented, where agents negotiate, hire, and pay for services from other agents through the centralizing entity. However, it does not allow each BDI agent to own a wallet. In addition, it is mandatory using an artificial institution (following the *Situated Artificial Institution* [5] model) that is recognized by all agents and other entities in the system because the notion of money transfer depends on the interpretation of each agent. Finally, the delay in effecting transactions is another limitation.

Minarsch et al. [14] presents a framework for *Autonomous Economic Agents* (AEA) to operate on Ethereum on behalf of an owner, with limited or no interference from that owner entity and whose objective is to generate economic value for its owner. Allows developers to distribute agents as finished products to end users, reducing barriers to widespread MAS adoption. However, AEA uses an abstraction based on behaviors and handler code. It does not have BDI-based fundamentals and does not support content based on ontology, agent persistence, and agent mobility services.

This work presents a middleware for integrating MAS with BigChainDB, a distributed database with blockchain characteristics. Unlike Calvaresi et al., this paper eliminates the need for a certifying authority. Each agent can generate a wallet to interact with the DLT. Unlike Papi et al., creating a virtual institution is unnecessary, and each agent can manage digital assets directly via new internal actions provided by the middleware. Finally, unlike Minarsch et al., this work allows BDI agents to generate economic value for their owners.

4 Proposed middleware

This paper presents some approaches for using divisible and indivisible assets as concordance factors in intelligent agents' relationships. New internal actions are proposed to integrate BDI agents with a DLT. In addition, as a proof of concept, a middleware was developed, enabling agents to create and transfer assets and to manage digital wallets in the permissioned blockchain BigchainDB.

4.1 Indivisible assets supporting the BDI agents' relationship

Agents can create and transfer indivisible assets that are unique and immutable records in the DLT, cryptographically signed and protected by crypto-conditions. With this, they can represent ownership registrations and transfers of artifacts, publicize beliefs and plans, or even record promises and commitments. Below are some possible approaches for using indivisible assets by intelligent agents.

Indivisible asset such as property record: Artifacts provide some function or service for agents to achieve their goals. An artifact can be a computational device that populates the MAS's environment dimension [20] or a physical device capable of acting or perceiving the real world [12]. An artifact can be used individually or collectively by agents. So, DLT can add a layer of ownership to artifacts, making it easier to implement access control and security. Using indivisible assets as a property record, the artifact itself can use the DLT as a reliable basis for defining permissions, always consulting the last transaction of the asset.

Figure 1 presents an example where agent Bob registers an asset in the DLT, representing an artifact. Subsequently, it transfers ownership to agent Alice through a transaction. Then, agent Alice transfers to agent Eve, the current owner. The asset's immutable characteristics are recorded at creation: the artifact name and a serial number. The asset has metadata that is added to each new transaction.

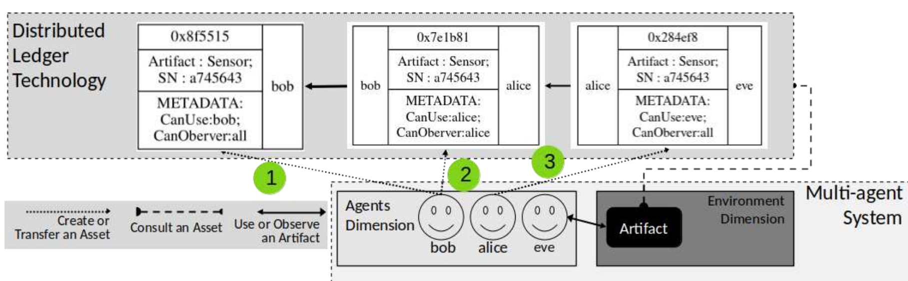

Fig. 1: Using indivisible asset such as property record.

When creating the asset (1), the metadata represents that the artifact can be observed by all agents and used only by agent Bob. When the asset was transferred to agent Alice (2), the artifact could only be observed and used by agent Alice. Finally, in the last transaction (3), the artifact can be observed by all agents and used only by agent Eve, the current owner of the artifact.

Indivisible asset such as promise or agreement: By analyzing the results of interactions between agents or information received from other agents, trust models seek to guide how, when, and with which agents it is safe to interact [19]. DLT technologies can add a layer of trust to the relationship between intelligent agents. In this case, a history of an agent's reputation can is built through an asset. Any agent that receives the asset will be able to analyze the history of agreements or promises made by the agent.

Figure 2 presents an example of agent bob committing to a particular task. After creating the asset (1) representing the commitment, the agent transfers (2) it to agent alice. When agent Bob fulfills his commitment and agent Alice is satisfied, he returns the asset (3) to Bob. Later, agent bob promised to eve can assume the same commitment (4).

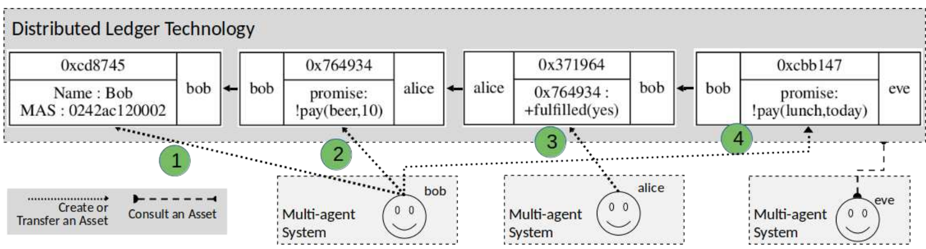

Fig. 2: Using invisible asset such as promise.

4.2 Divisible assets to support the agents' relationship

A divisible asset can represent a cryptocurrency in DLT, so that can be created several tokens. All divisible assets are created in a wallet, are cryptographically signed, and initially, the wallet concentrates all the asset units, being able to transfer units of this asset to other wallets [13]. Intelligent agents can use divisible assets to trade with other agents inside or outside the MAS. Figure 3 presents

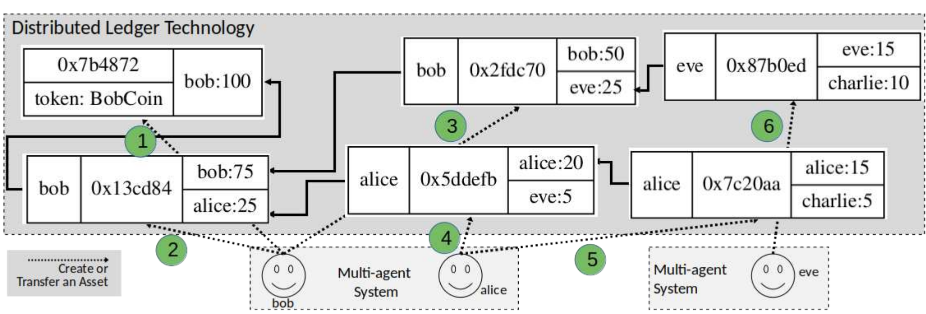

Fig. 3: Using divisible assets to support the agents' relationship

an example of a transaction involving divisible assets. Agent Bob creates 100 units of the asset BobCoin (1) and performs two transfers: in the first (2), 25 Bobcoins are sent to agent alice; in the second (3), another 25 BobCoins are sent to agent eve. Agent Alice, in turn, transfers 5 Bobcoins to agent eve (4) and another 5 Bobcoins to agent charlie (5). Likewise, agent eve transfers 10 Bobcoins to agent charlie (6). Finally, what remained: agent bob has 50; agent alice has 15; agent eve has 20; agent charle has 15; totaling the 100 Bobcoins created and distributed by DLT.

4.3 Stamping a transaction.

In a negotiation scenario between agents, a producer, and a consumer, the consumer agent transfers an asset to the producer agent, requesting a specific service. In turn, the producing agent verifies the transaction's validity on the DLT. Once the transaction is valid, the agent executes the service. In a typical scenario, the producer agent should store in its belief base or, worst case, in a database where a specific request has already fulfilled a specific transaction. It is necessary to prevent a malicious agent from requesting a service, using the same transaction several times.

A transaction is considered open if the OUTPUT pointer does not point to the INPUT pointer of the next transaction. It represents in this way that the transaction has not yet been spent. In addition to implementing internal actions to operate in DLT from the agents' dimension, this paper also presents the concept of stamping a transaction.

Definition 1 (Stamp Transaction). *Stamping a transaction is a self-transfer and unification process. Self-transfer, therefore, the units received from a divisible asset are transferred to itself, spending the received transaction (filling the OUTPUT pointer with the address of its wallet). Unification because this process joins the units from the received transaction with those already in the wallet. A transaction with two or more INPUT pointers and a single OUTPUT pointer is generated in this process.*

4.4 Middleware Velluscinum

Middleware Velluscinum[3] extends the jason-lang [3] through integration with BigchainDB [13], providing new internal actions to enable the use of digital assets to support the relationship between intelligent agents. Figure 4 presents the integration of two MAS with a DLT. The actions offered by the middleware are available directly to the dimension of the agents that populate the MAS. They bridge the Multi-agent Oriented Programming (MAOP) paradigm [1] and the BigchainDB communication driver [13].

In this way, intelligent agents can create or transfer digital assets, stamp transactions or manage their wallets on the DLT directly from their dimension.

[3] https://velluscinum.chon.group/

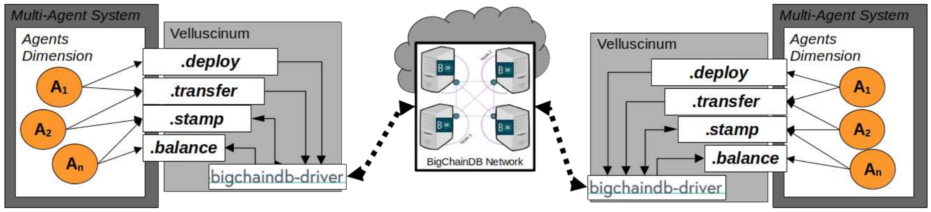

Fig. 4: Proposed middleware approach

The built-in internal actions provided by the middleware are described below:

– **.buildWallet**(w) - generates a digital wallet and returns the belief +w(P,Q);
– **.deployNFT**(S,P,Q,I,M,b) - registers an asset and returns the belief +b(A);
– **.transferNFT**(S,P,Q,A,R,M,b) - transfer an asset and returns +b(T);
– **.deployToken**(S,P,Q,I,V,b) - creates V units from an asset, returns +b(C);
– **.transferToken**(S,P,Q,C,R,V,b) - transfer V units of C and returns +b(T);
– **.stampTransaction**(S,P,Q,T) - stamps a transaction (T);
– **.tokenBalance**(S,P,Q,C,q) - check the wallet Q and return +q(C,V).

Where:

– b is a belief that represents a result of an operation in DLT;
– w is a belief that represents an agent's wallet;
– q is a belief that represents the balance of C in the agent's wallet.
– A is a literal that represents a divisible asset;
– C is a literal that represents a indivisible asset;
– P e Q are literals that represent the agent's key pair;
– R is a literal that represents the public key of a recipient agent;
– S is a literal that represents the address of a DLT node;
– T is a literal that represents a transaction performed in the DTL;
– V is a literal that represents the number of parts of a C;
– I is a key-value array that represents the immutable data of an asset;
– M is a key-value array representing asset or transaction metadata;

5 Case Study

To validate the proposed approach, we present the adaptation of a well-known example of relationships between intelligent agents for the use of digital assets. The Building-a-House example [1] presents a multi-agent system scenario with an interorganizational workflow for building a house on the land of a contracting agent. The agent hires builder agents during an auction to achieve this overall objective. Furthermore, coordination is needed to carry out the tasks related to the construction of the property.

The original example uses artifacts to manage the auction for each stage of construction. In this integration, we use the approach of indivisible assets as an agreement. Thus, before creating artifacts, added plans to deploy an asset for each stage of construction It represents a contract that is transferred to the winner before the execution of the task, and returned to the owner after the payment confirmation.

Figure 5 presents the necessary adaptations for all agents and the specific adaptations for the owner agent to integrate the example with the proposed approach. The adaptations are detailed below: Before the execution of the MAS, a digital currency (JacamoCoin) and a wallet for the owner agent are created. In addition, currency units are transferred to the owner's wallet; In the source code common to all agents in the system, a belief is added containing JacamoCoin's ASSET-ID and the address of a DLT node (common.asl, lines 2-3). This way, when starting the MAS, the agent already has a balance, and all agents agree with the cryptocurrency in that MAS; A belief is added to the owner agent, containing its wallet's private and public key (giacomo.asl, lines 2-3); In the creation plan of the artifact responsible for the auction were added actions to generate an asset representing a contract referring to the task to be auctioned (giacomo.asl, lines 50-56); In the auction result display plan was added actions to request the information necessary to transfer the digital asset to the winner (giacomo.asl, lines 71-75); In the contract execution plan, was added information about the digital asset in the message sent to the winners (giacomo.asl, lines 126-129); A plan responsible for carrying out the transfer of the digital asset that represents the contract between the owner and the contractor was added (giacomo.asl, lines 144-152); Finally, a plan to carry out the payment of the task after its execution by the contractor was added (giacomo.asl, lines 155-164).

The owner agent initiates the execution phase of the house construction project by requesting the winners to carry out the tasks. At this stage, organizations verify the validity of the asset representing the contract in the DLT. This action is performed through the transaction stamp. Once the contract is confirmed, the company starts executing the task. Upon completion of the execution, as defined in the contract, the company requests payment in JacamoCoins (digital currency accepted in the SMA - divisible assets approach to support the relationship between agents). Finally, when the agent informs about the payment of the task, the company confirms receipt (transaction stamp approach). If everything is correct, the company returns the asset representing the contract to the owner.

Figure 6 presents the adaptations necessary for all organizations that enable the integration of the example with the proposed approach. The adaptations are detailed below: An action was added to request payment after executing each auction task (org_goals.asl, lines 5;8;10;12;15; 17;19;21); The necessary information for triggering the contract execution plan has been changed. In addition, before executing the tasks, a plan for validating the contract with the DLT is activated (org_code.asl, lines 13 and 15); A plan was added to provide for the creation of a virtual company wallet, along with the DLT (org_code.asl, lines 40-45); A plan was added to note in the agent's mind the ASSET-ID of the contract he won in the auction and also to inform the owner which wallet will

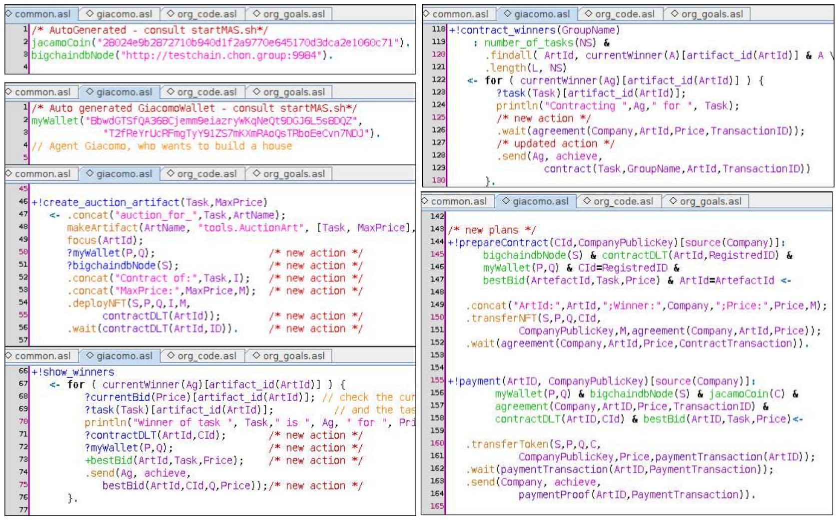

Fig. 5: Generic and specific changes to the giacomo agent to adhere to the Building-a-House[1] example with the proposal of digital assets as support for the relationship between intelligent agents.

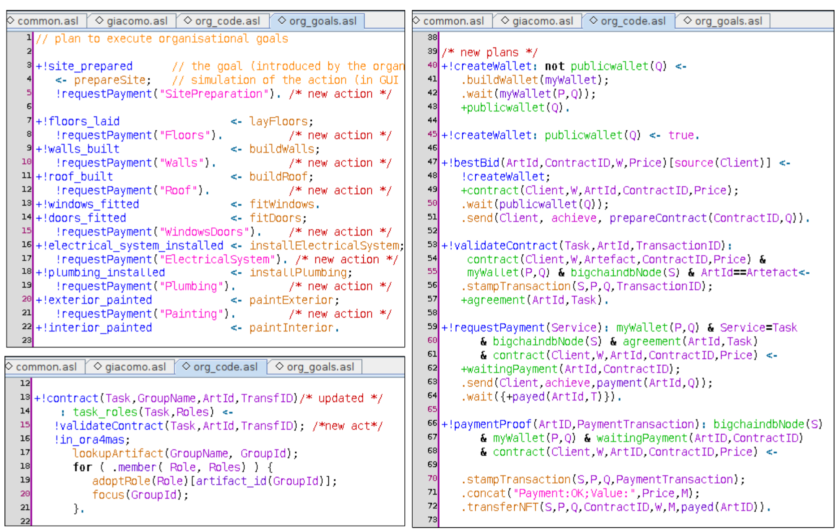

Fig. 6: Generic changes were needed by organizations to adhere to the Building-a-House [1] example with the proposal of digital assets to support the relationship between intelligent agents.

receive the contract (org_code.asl, lines 47-51); A plan was added with the necessary actions to validate contracts with the DLT (org_code.asl, lines 53-57); A plan has been added to make it possible to request payment after executing a task (org_code.asl, lines 59-64); Finally, a plan was added to validate a payment and transfer ownership of the contract to the owner (org_code.asl, lines 66-72).

6 Conclusion

This paper presents an approach for using digital assets in the relationships between cognitive agents, enabling the representation of the transfer of funds, registration of property, declaration of promises, and dissemination of knowledge. Integration of a well-known example of MAS with DLT was presented using middleware for the agents' dimension. In addition, it was possible to: evaluate the functioning of the system and verify that each agent can sign its transaction through its asymmetric key pair; use the DLT as an open and reliable basis to feed agents' beliefs; and enable the manipulation of assets directly by the agents. Future works can analyze the need for new internal actions for the agent dimension, more complex scenarios involving transactions between different multi-agent systems, and the possibilities and implications of a specific permissive DLT for intelligent agents.

References

1. Boissier, O., Bordini, R.H., Hübner, J.F., Ricci, A., Santi, A.: Multi-agent oriented programming with jacamo. Science of Computer Programming **78**(6), 747–761 (2013). https://doi.org/10.1016/j.scico.2011.10.004
2. Boissier, O., Hübner, J.F., Ricci, A.: The JaCaMo Framework, pp. 125–151. Springer International Publishing, Cham (2016). https://doi.org/10.1007/978-3-319-33570-4_7
3. Bordini, R., Hübner, J., Wooldridge, M.: Programming Multi-Agent Systems in AgentSpeak using Jason. Wiley Series in Agent Technology, Wiley (2007)
4. Bratman, M.: Intention, Plans, and Practical Reason. Cambridge: Cambridge, MA: Harvard University Press (1987)
5. de Brito, M., Hübner, J.F., Boissier, O.: Situated artificial institutions: stability, consistency, and flexibility in the regulation of agent societies. Autonomous Agents and Multi-Agent Systems **32**(2), 219–251 (2018). https://doi.org/10.1007/s10458-017-9379-3
6. Calvaresi, D., Calbimonte, J.P., Dubovitskaya, A., Mattioli, V., Piguet, J.G., Schumacher, M.: The good, the bad, and the ethical implications of bridging blockchain and multi-agent systems. Information **10**(12) (2019). https://doi.org/10.3390/info10120363
7. Contreras, A.F.: Benchmarking of blockchain technologies used in a decentralized data marketplace. Grado en Ingeniería Informática, E.T.S de Ingenieros Informáticos (UPM), Madrid, España (Jun 2019), https://oa.upm.es/55775/
8. Dabbagh, M., Choo, K.K.R., Beheshti, A., Tahir, M., Safa, N.S.: A survey of empirical performance evaluation of permissioned blockchain platforms: Challenges and opportunities. Computers Security **100**, 102078 (2021). https://doi.org/10.1016/j.cose.2020.102078
9. El Ioini, N., Pahl, C.: A review of distributed ledger technologies. In: On the Move to Meaningful Internet Systems. OTM 2018 Conferences. pp. 277–288. Springer International Publishing, Cham (2018)

10. Ge, Z., Loghin, D., Ooi, B.C., Ruan, P., Wang, T.: Hybrid blockchain database systems: Design and performance. Proc. VLDB Endow. **15**(5), 1092–1104 (may 2022). https://doi.org/10.14778/3510397.3510406
11. Kampik, T., Najjar, A.: Simulating, off-chain and on-chain: Agent-based simulations in cross-organizational business processes. Information **11**(1) (2020). https://doi.org/10.3390/info11010034
12. Manoel, F., Pantoja, C.E., Samyn, L., de Jesus, V.S.: Physical Artifacts for Agents in a Cyber-Physical System: A Case Study in Oil & Gas Scenario (EEAS). In: The 32nd International Conference on Software Engineering and Knowledge Engineering, SEKE 2020. pp. 55–60. KSI Research Inc. (2020)
13. McConaghy, T., Marques, R., Müller, A., De Jonghe, D., McConaghy, T., McMullen, G., Henderson, R., Bellemare, S., Granzotto, A.: Bigchaindb: a scalable blockchain database. white paper, BigChainDB (2016)
14. Minarsch, D., Favorito, M., Hosseini, S.A., Turchenkov, Y., Ward, J.: Autonomous economic agent framework. In: Alechina, N., Baldoni, M., Logan, B. (eds.) Engineering Multi-Agent Systems. pp. 237–253. Springer International Publishing, Cham (2022). https://doi.org/10.1007/978-3-030-97457-2_14
15. Nasir, Q., Qasse, I.A., Abu Talib, M., Nassif, A.B.: Performance Analysis of Hyperledger Fabric Platforms. Security and Communication Networks **2018**, 1–14 (2018). https://doi.org/10.1155/2018/3976093
16. Papi, F.G., Hübner, J.F., de Brito, M.: A blockchain integration to support transactions of assets in multi-agent systems. Engineering Applications of Artificial Intelligence **107**, 104534 (2022). https://doi.org/10.1016/j.engappai.2021.104534
17. Podgorelec, B., Turkanović, M., Šestak, M.: A brief review of database solutions used within blockchain platforms. In: Blockchain and Applications. pp. 121–130. Springer International Publishing, Cham (2020)
18. Rajasekaran, A.S., Azees, M., Al-Turjman, F.: A comprehensive survey on blockchain technology. Sustainable Energy Technologies and Assessments **52**, 102039 (2022). https://doi.org/10.1016/j.seta.2022.102039
19. Ramchurn, S.D., Huynh, D., Jennings, N.R.: Trust in multi-agent systems. The Knowledge Engineering Review **19**(1), 1–25 (2004)
20. Ricci, A., Viroli, M., Omicini, A.: Programming mas with artifacts. In: Bordini, R.H., Dastani, M.M., Dix, J., El Fallah Seghrouchni, A. (eds.) Programming Multi-Agent Systems. pp. 206–221. Springer Berlin Heidelberg, Berlin, Heidelberg (2006)
21. Ruan, P., Dinh, T.T.A., Loghin, D., Zhang, M., Chen, G., Lin, Q., Ooi, B.C.: Blockchains vs. distributed databases: Dichotomy and fusion. In: Proceedings of the 2021 International Conference on Management of Data. p. 1504–1517. SIGMOD '21, New York, NY, USA (2021). https://doi.org/10.1145/3448016.3452789
22. Shalaby, S., Abdellatif, A.A., Al-Ali, A., Mohamed, A., Erbad, A., Guizani, M.: Performance Evaluation of Hyperledger Fabric. In: 2020 IEEE International Conference on Informatics, IoT, and Enabling Technologies (ICIoT). pp. 608–613. IEEE, Doha, Qatar (Feb 2020). https://doi.org/10.1109/ICIoT48696.2020.9089614
23. Sichman, J.S., Demazeau, Y., Boissier, O.: When can knowledge-based systems be called agents. In: Simpósio Brasileiro de Inteligência Artificial. vol. 9, pp. 172–185. SBC, Rio de Janeiro (1992)
24. Tikhomirov, S.: Ethereum: State of knowledge and research perspectives. In: Imine, A., Fernandez, J.M., Marion, J.Y., Logrippo, L., Garcia-Alfaro, J. (eds.) Foundations and Practice of Security. pp. 206–221. Springer International Publishing, Cham (2018). https://doi.org/10.1007/978-3-319-75650-9_14
25. Wooldridge, M.J.: Reasoning about rational agents. MIT press (2000)