STM JOURNALS

Journal of
Artificial Intelligence
Research & Advances

ISSN: 2395-6720
Volume 13, Issue 1, 2026
January-April
DOI Journal: 10.37591/JoAIRA

https://journals.stmjournals.com/joaira

Research JoAIRA

Autonomous Agent Contracts for Adaptive Supply Chains

Bhargav Chebrolu*

Abstract

This paper proposes a novel agent-driven, on-chain coordination layer for manufacturer–distributor–retailer handoffs that automates release, transfer, receipt, and state validation across the supply network. This approach combines belief–desire–intention (BDI) agents with narrowly scoped smart contracts to capture role responsibilities, capacity checks, and escalation policies. It aims to enhance conformance, traceability, and cycle-time reliability without relying on central intermediaries. This proof of concept links supply chain states—such as production-ready, packaged, listed, purchased, shipped, and delivered—to contract primitives, while BDI agents trigger transactions based on their beliefs regarding inventory and production readiness. Auditable workflows and dispute-resilient confirmations are thereby established. The implementation artifacts produced by the project include Jason-based multi-agent behaviors, Solidity contracts for item lifecycle management and ownership, and deployment and execution cost measurements to demonstrate operational feasibility at the retail distribution edge. The results demonstrate that autonomous governance mechanisms can synchronize material flows, control handoff events, and surface governance signals, thereby providing a practical pathway for embedding accountable decision logic into supply chain processes.

Keywords: Autonomous systems, BDI agents, blockchain, multi-agent systems, smart contracts, supply chain management

INTRODUCTION

Modern supply chains face significant challenges in terms of coordination, transparency, and trust among the participants. Traditional centralized systems suffer from single points of failure, a lack of verifiable audit trails, and limited adaptability to dynamic market conditions [1]. The emergence of blockchain technology and smart contracts offers promising solutions to these challenges by providing decentralized tamperproof ledgers and automated contract execution [2].

Concurrently, belief–desire–intention (BDI) agents have demonstrated remarkable capabilities in autonomous decision-making and adaptive behavior in complex environments [3]. These software agents operate based on their beliefs about the environment, desires to represent their goals, and intentions to represent committed courses of action. The integration of BDI agents with blockchain-based smart contracts presents a novel approach for creating intelligent decentralized supply chain systems that can autonomously coordinate activities while maintaining transparency and accountability. This study addresses the critical need for automated coordination mechanisms in supply chain management by developing a framework that combines Jason BDI agents with Ethereum smart contracts. The proposed system enables manufacturer, distributor, and retailer handoffs to be automatically executed and verified on-chain coordination layer while maintaining the flexibility and adaptability characteristics of

***Author for Correspondence**
Bhargav Chebrolu
E-mail: bhargav.cheb@gmail.com

Research Scholar, MS in Supply Chain Management (Naveen Jindal School of Management), The University of Texas, Dallas, Richardson, TX 75080, United States

Received Date: December 29, 2025
Accepted Date: December 31, 2025
Published Date: January 15, 2026

Citation: Bhargav Chebrolu. Autonomous Agent Contracts for Adaptive Supply Chains. Journal of Artificial Intelligence Research & Advances. 2026; 13(1): 52–60p.

intelligent agent systems. This approach focuses on encoding supply chain states as contract primitives and using BDI agents to trigger transactions based on their beliefs about inventory levels and production readiness. The contributions of this study are as follows.

- A novel architecture for integrating BDI agents with smart contracts in supply chain contexts,
- Implementation of role-specific behaviors for supply chain participants using the Jason framework,
- Development of Solidity smart contracts for item lifecycle management, and
- Empirical evaluation of deployment and execution costs to assess operational feasibility.

The system demonstrates how autonomous policies can synchronize material flows, enforce handoffs, and provide governance events, offering a practical path toward embedding accountable decision logic in supply chain processes.

BACKGROUND AND RELATED WORK

Belief–Desire–Intention Agents

The BDI model is a well-known architecture for creating intelligent agents that can act independently in changing environments, and it has three important mental attitudes [4]: beliefs (i.e., their knowledge about the environment), desires (which are their goals), and intentions (which are the plans they will execute to achieve their goals). This architecture helps agents balance their reactive behavior with goal-driven behavior. As such, they are useful in domains such as supply chain management. AgentSpeak(L) is an abstract programming language based on the BDI architecture with formal semantics for agent programming [5].

Jason refers to an extended interpreter for AgentSpeak(L) that allows the development of multi-agent systems with a wide range of customization options. The structure of the model supports inter-agent communication based on speech acts, strong negation, and distributed deployment on infrastructure such as JADE [6]. Alternatives to BDI frameworks include ASTRA, which resembles Java in its typing and syntax [7], and JADEX, which merges BDI reasoning with the JADE middleware [8].

Blockchain and Smart Contracts

Blockchain technology offers the establishment of decentralized ledgers and immutability that can record transactions without trusted third parties [9].

Wu [2] first conceptualized smart contracts. These are contracts that execute themselves with terms written in the code. Ethereum made smart contracts popular and introduced a turning complete virtual machine to execute them [10]. Smart contracts in supply chain contexts enable automated execution of business logic, transparent tracking of goods, and immutable recording of transactions. Several researchers have explored blockchain applications in supply chains by focusing on traceability, provenance, and automated payments [11]. However, most existing approaches lack the adaptive decision-making capabilities provided by BDI agents.

Agent-Blockchain Integration

Recent research has investigated the potential of multi-agent systems and blockchain interoperability. Calvaresi et al. [12] identified patterns and issues using a systematic literature survey. Chen et al. proposed an agent model for decentralized execution of smart contracts to avoid manipulation [1]. Ciatto et al. provided an overview of the preparation and integration of agents and blockchains in both computational and interactional terms [13].

Earlier research concentrated on using blockchain as a trust infrastructure for agents or agents that manage the execution of smart contracts. Our approach differs from smart contract state transitions that are tightly coupled with BDI reasoning, allowing agents to automatically manage supply chain workflows while using blockchain for transparency and immutability (Table 1).

SYSTEM ARCHITECTURE AND DESIGN

Overall Architecture

The architecture of our proposed system consists of three segments: (1) a multi-agent system manifesting supply chain roles through BDI agents; (2) smart contracts in Ethereum managing the life cycle of items and transitions of ownership; and (3) a coordination layer enabling agent-blockchain communication. As a result, decisions can be made collectively, while every transaction remains transparent and accountable.

The four main agent types in the multi-agent system are the manufacturer agent, wholesaler agent, retailer agent, and supply chain agent that are coordinator agents. Each agent has beliefs about inventory levels, production capacity, and business relations.

Desires reflect business goals, such as maintaining optimal inventory levels or fulfilling customer orders. A specific set of efforts, such as production runs or orders to suppliers, is referred to as intention (Figure 1).

The storage operations were maintained to a minimum. Check-effects-interaction patterns are implemented in contracts to avoid reentrancy attacks.

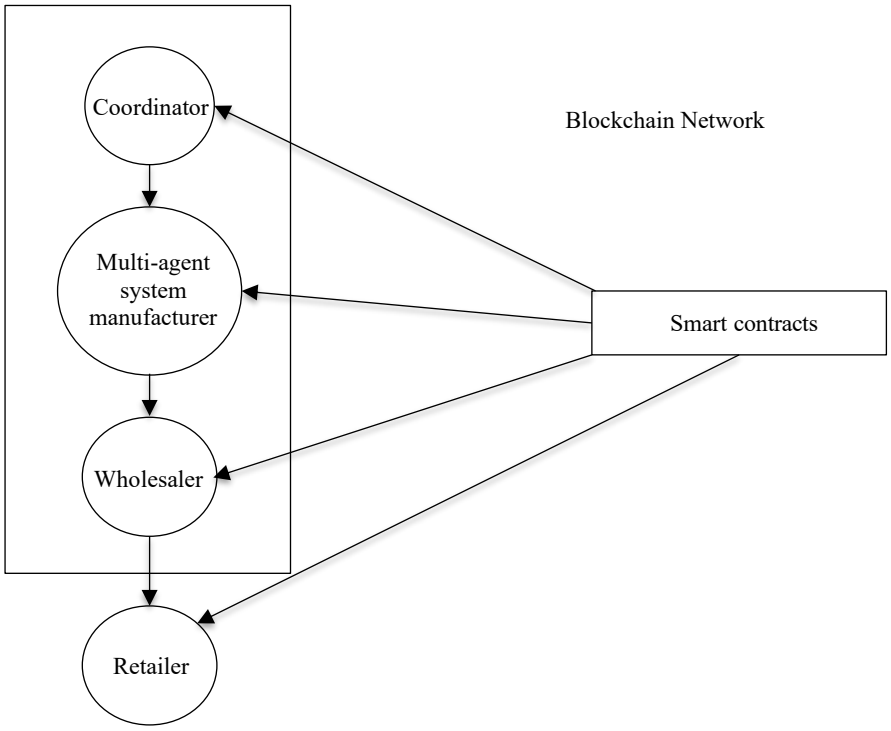

Figure 1. System architecture overview.

Table 1. Comparison of BDI agent frameworks.

Framework	Language	Key features
Jason (Java)	AgentSpeak, Java	Concurrent programming, speech-act communication, full AgentSpeak interpreter
Jason (Python)	AgentSpeak, Python	Event-based programming, speech-act communication, Python integration
ASTRA	Java-like syntax	Strong typing, Java integration, and plan bodies for code reuse
JADEX	Java	JADE extension, rational agents, web service focus
LightJason	AgentSpeak(L++), Java	Lambda expressions, parallel execution, thread-safe variables
JaCaMo	AgentSpeak, Jason	Combines CArtAgO, Moise, and Jason for comprehensive Multi-Agent System (MAS) development

Smart Contract Design

Supply chain smart contracts use business logic that applies to operations. Smart contracts are applied to map the states of the physical supply chain to the representations of the chain. The functions of the contract include the following.

- produceItemByManufacturer,
- packageItemByManufacturer,
- sellItemByManufacturer,
- purchaseItemByWholesaler,
- shippedItemByManufacturer,
- receivedItemByWholesaler,
- sellItemByWholesaler,
- purchaseItemByRetailer,
- shippedItemByWholesaler,
- receivedItemByRetailer.

Each function maps to a single state transition in the supply chain workflow. In addition, these functions perform appropriate checks to verify compliance with business rules. The contracts have state variables that track who owns the item, where it is, and what condition it is in. An event is emitted for each state transition, such that all actions can be audited.

The contract design is in line with the best practices for security and gas optimization. We used function modifiers for access control. No state change is permitted in the view functions.

Agent Behavior Modeling

The smart decision-making logic of BDI agents helps supply chains. The beliefs, desires, and intentions of a type are supplied for their roles in the supply chain.

The manufacturer considers how much they can produce, how much raw material they have, and how many orders they are waiting to fill. The main aim is to determine the optimal production scheduling that meets demand and minimizes costs. The main aims of production planning are to start production when stocks cross certain threshold levels, package finished products, and ship them to wholesalers.

The WholesalerAgent tracks stock levels, sales trends, and manufacturer lead-times. It aims to maintain stock at balanced levels, regardless of the retailer's needs. Companies place orders with manufacturers when they start to run low on inventory. Subsequently, the manufacturer's shipment is received into the inventory, and the retailer's order is filled and shipped.

The RetailerAgent observes the rank of demand, stock on shelves, and wholesaler availability. This is an efficient process aimed at fulfilling the customer's requirements. The intent is to place orders with wholesalers, receive shipments, and update the records. SupplyChainAgent ensures the smooth functioning of the supply chain by coordinating different activities between agents. It wants to run the supply chain as efficiently as possible and solve disruptions. Intentions involve starting supply chain processes, raising issues when necessary, and creating performance reports.

IMPLEMENTATION DETAILS

Multi-Agent System Development

The multi-agent system was implemented using the Jason framework with both Java and Python interpreters. The Java-based implementation utilized Jason version 3.1 with Gradle build automation, whereas the Python implementation used the AgentSpeak package (version 0.1.0) installed via pip. Both implementations followed the same architectural patterns but differed in their integration approaches with the blockchain layer.

Each agent was programmed in AgentSpeak(L) with. asl files, defining their beliefs, goals, and plans. The manufacturer includes plans for production initiation, quality control, and shipment preparation. WholesalerAgent implemented plans for inventory management, order processing, and logistics coordination. RetailerAgent contains plans for demand forecasting, shelf management, and customer service. The SupplyChainAgent coordinated activities across all agents and handled the exception scenarios.

Agent communication followed FIPA ACL standards, with messages containing performatives such as requests, forms, and proposals. Agents maintain conversation states to manage multi-step interactions and implement timeout mechanisms for robust communication. The system supports both synchronous and asynchronous message passing with agents capable of handling multiple concurrent conversations.

Smart Contract Development

Smart contracts were developed using Solidity version 0.8.13 Truffle Suite (version 5.6.5) for compilation, testing, and deployment. The ERC-1155 multi-token standard was used for efficient management of product types and batches in the contract design. The Universal Product Code (UPC) and stock-keeping unit (SKU) were placed on each product for traceability in the supply chain.

The implementation of the contract requires an access control mechanism using OpenZeppelin libraries that assign role-based permissions to manufacturers, wholesalers, and retailers. The use of a modifier checks the enforced state transitions that ensure that actions are taken in the correct sequence, for example, a produced product cannot be shipped. All the state changes emitted events that could be followed by any off-chain system deployed in the supply chain.

The testing was performed in Truffle using JavaScript. The tests were unit tests on functions and integration tests for workflows. To reduce gas usage, the data types must be properly selected. Storage operations must be less operational. Batch processing must be performed whenever possible. The contracts were deployed to one local test network (Ganache) and two public test nets (Alfajores and Rinkeby).

Integration Approach

We were able to integrate BDI agents with smart contracts using Web3 libraries that offer pragmatic access to the Ethereum blockchain. The Python-based implementation of Jason used Web3.py to connect to Ethereum's nodes and interact with the deployed contracts. Web3j (version 4.9.4) offers related functionality for Java-based implementation.

The Jason framework was used to define custom actions for interacting with smart contracts. These actions were set as Python functions (for the Python interpreter) or Java methods (for the Java interpreter) that used Web3 libraries to trigger the contract functions. The actions managed transaction signing, estimating gas, and verifying receipts, offering an effortless interface for agents to connect to the blockchain.

The integration supported both the read and write operations. The system executes read and write operations in different ways. It reads operations synchronously. In contrast, it performed write operations asynchronously. If a transaction fails, the system attempts the transaction again. In addition, if the network goes down, we can attempt another option.

EXPERIMENTAL EVALUATION

Experimental Setup

We performed the assessment in a local testing network called Ganache version 7.5.0, which acts like an Ethereum blockchain. The actor instances include the following:

- ManufacturerAgent,

- WholesalerAgent,
- RetailerAgent, and
- SupplyChainAgent that ran on Ubuntu 22.04.1 LTS.

Java SE Development Kit 18.0.2.1 (JDK 18.0.2.1) and Python 3.8.10. The network was set up so that components had minimal latency, so we could focus on system performance and not network effects.

The test scenario simultaneously covered various supply chain events, including normal operations, inventory shortages, production delays, and multiple events at the same time. Multiple iterations were performed for each scenario to account for the variance in blockchain processing times and agent decision cycles. The parameters collected to evaluate the performance included action confirmation times, gas consumption, agent response times, and system throughput.

We compared three different integration methods: Jason with the Java interpreter and Web3j, Jason with the Python interpreter and Web3.py, and the ASTRA framework with Web3j for Ethereum 2.0. All approaches were evaluated based on development complexity, runtime efficiency, and integration stability (Figure 2).

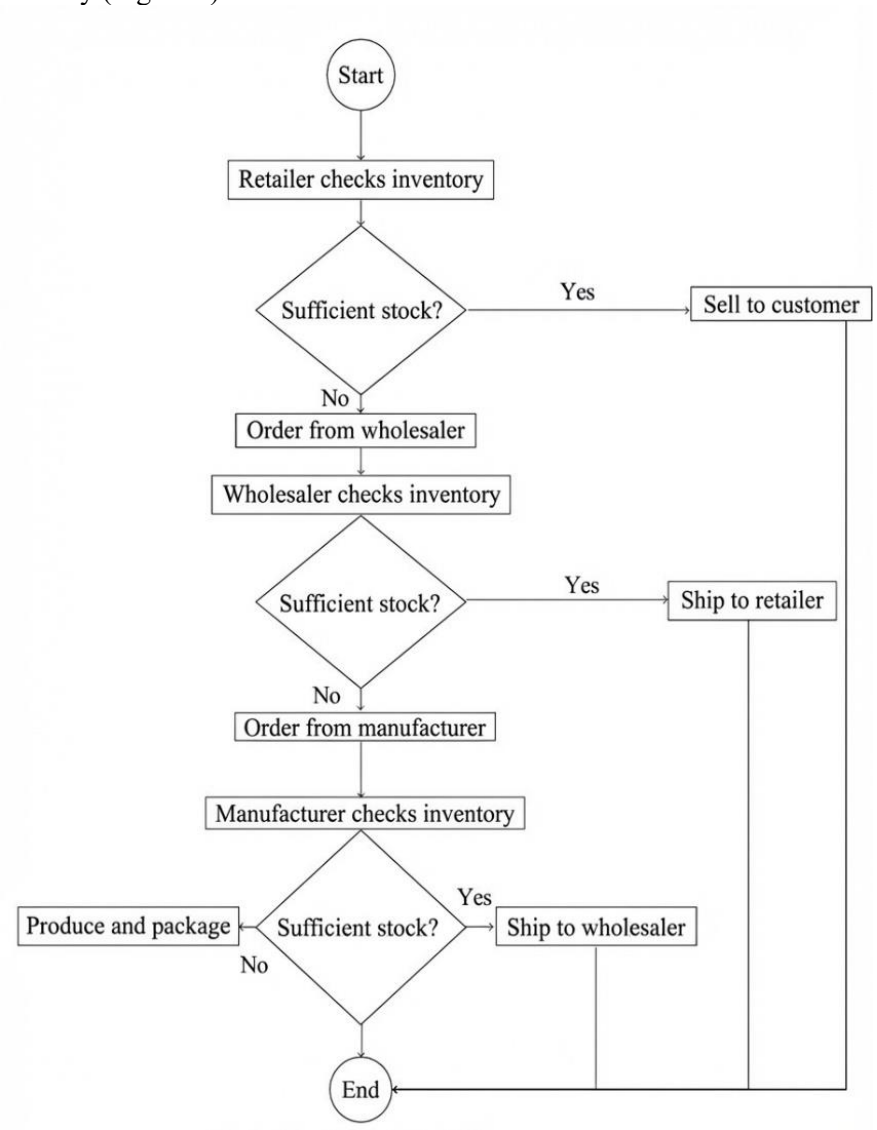

Figure 2. Supply chain workflow.

Performance Analysis

There was a consistent gas consumption analysis across supply chain operations. The highest consumption of gas (approximately 85,000–95,000) was due to product manufacturing functions that had many state updates and emission events. The shipping and receiving operations required moderate amounts of gas, and the operations required 45,000–60,000 gas units. By contrast, making a simple query to the state went through quickly at a gas cost of 21,000 units.

The time taken for a transaction to be confirmed varies according to the network conditions and gas prices. Transactions are typically confirmed in 2–3 s on a local test network with low congestion. During times of high network activity, public testnets also experienced increased confirmation times of 15–30 s. The system exhibited stable performance under normal operational conditions. It can handle five to seven transactions per minute for typical supply chain processes. A majority of the agent decision cycles occurred in the range of 100–500 ms, whereas the more complicated planning scenarios took approximately 2 s. The response times of the Jason implementation in Python were slightly faster than those in Java; however, both were sufficiently fast for the supply chain. The performance of ASTRA is similar to that of other models, such as Jason with Java, but requires more configuration (Table 2).

Qualitative Assessment

Supply chain management can gain various qualitative benefits through integrated system usage. Keeping all transactions and state changes immutable increases a high level of transparency. The participant can verify all event history, such as supply chain, production, ownership changes, shipping events, etc.

In addition, through automation, repetitive processes are performed without manual effort, as agents manage inventory and are shipped independently. This accelerated things up and decreased the chances of errors. Owing to its flexibility, the system can auto-implement contingency plans if there is an inventory shortage or a production problem.

The system was not reliant on a central authority, as there was no single point of failure. Cryptography is used to allow participants to trade directly. It enables trust. Agents can behave flexibly and in a controlled manner by using BDI reasoning based on blockchain execution. These actions have accountability properties and comply with the business rules.

DISCUSSION AND FUTURE WORK

Research Implications

This study shows how BDI agents can be integrated into blockchain smart contracts to manage supply chains. This approach solves the most important problems that persist in current supply chain systems. Issues such as a lack of transparency and manual coordination processes. This means that the social accountability and adaptation of agents can be achieved by encoding social accountability in a smart contract and in an agent's plan.

Table 2. Gas consumption by smart contract function.

Contract function	Gas consumed
produceItemByManufacturer	92,145
packageItemByManufacturer	47,832
sellItemByManufacturer	58,419
purchaseItemByWholesaler	52,763
shippedItemByManufacturer	45,228
receivedItemByWholesaler	43,915
sellItemByWholesaler	56,894
purchaseItemByRetailer	51,672
shippedItemByWholesaler	44,583
receivedItemByRetailer	42,719

The findings help increase research on how to integrate agents and blockchains into supply chains. Using the Jason framework, the implementation of behavior specific to the role shows that BDI agents can manage complex supply chain operations involving blockchain interaction effectively. By comparing integration techniques, developers can obtain an idea of how to combine them.

The study also highlights the need for suitable abstract layers between agents and the blockchain. The custom actions created for Jason agents can safely encapsulate the complexity of blockchain transactions to enable agent programmers to focus on business logic instead of blockchain complexities. This division of concerns helps to keep the code efficient, and processes are less complicated when developing the software.

Limitations and Challenges

Several limitations were identified during the research. The Java-based Jason implementation had trouble with the Web3j library and had to be configured a lot. Compatibility issues must be considered when using other technologies in conjunction with libraries. Although it is easier to implement in Python, it does not have some advanced features available in Java.

While we can manage the gas costs ourselves in the testing setup, it can become substantial in production, where there are high transaction volumes. For large-scale deployment, optimization techniques such as state channels or layer-2 solutions may be required. The underlying blockchain network conditions affect the system performance, which may vary significantly.

The current approach focuses on a simplified supply chain consisting of three main roles. In reality, supply chains are often composed of more intermediaries. Additional efforts are required to develop and validate the proposed approach for these scenarios.

Future Research Directions

This research opens up many possibilities for future research. Researching different BDI frameworks, such as LightJason or JADEX, is a possible solution. An examination of the interoperability with other blockchains, such as Hyperledger Fabric or Corda, may offer other tradeoffs in scalability, privacy, and governance.

If the Internet of Things (IoT) devices are used within the system, it will be possible to track a physical item at all times and create a digital twin. The agents would use the sensor data to make decisions at that time. The use of machine learning techniques will improve agent decisions. Demand forecasting, inventory optimization, and risk management are important.

Further research on scalability is required. Handling thousands of participants and many transactions in the supply chain. Various sharding techniques, side chains, and off-chain computation methods can help overcome scalability problems.

CONCLUSION

This study introduced a new supply chain approach that combines BDI agents and blockchain smart contracts. This system allows manufacturers, distributors, and retailers to handoff without interference but with full compliance, clarity, and ability to adapt. By using the Jason framework and Solidity smart contracts, the implementation shows the technical viability of the method and acceptable performance. Combination of BDI reasoning and blockchain execution can help to solve supply chain problems. Smart contracts adhere to business rules, offer immutable audit trails, and check BDI agents for adaptive decision-making and complex coordination. The harmony of the nonlinear dynamics of our organizational spheres enables contemporary control systems that are accountable and flexible. These systems can manage complicated and dynamic supply chain conditions. The tests proved that the system worked properly, and the gas consumption rates and performance characteristics were adequate. A comparison of integration methods provides important

information on how to mix and integrate technologies. There are still other challenges, such as scaling and real-world complexity, with regard to effectively using agent-blockchain systems for supply chain management. The method in this paper is to move toward smarter and more effective supply chains. When accountable decision logic is built into supply chain processes, organizations can achieve better coordination, efficiency, and customer satisfaction. The advanced growth of blockchain and agent technologies promises to change the way supply chains function in an increasingly complex world.

REFERENCES

1. Chen L, Xu L, Shah N, Gao Z, Lu Y, Shi W. Decentralized execution of smart contracts: agent model perspective and its implications. In: Brenner M, Rohloff K, Bonneau J, Miller A, Ryan PYA, Teague V, et al., editors. Financial Cryptography and Data Security. Cham: Springer; 2017. p. 468–477. doi:10.1007/978-3-319-70278-0_29.
2. Wu G, Wang H, Lai X, Wang M, He D, Chan S. A comprehensive survey of smart contract security: state of the art and research directions. J Netw Comput Appl. 2024;226:103882. doi:10.1016/j.jnca.2024.103882.
3. Rao AS, Georgeff MP. Modeling rational agents within a BDI-architecture. In: Allen J, Fikes R, Sandewall E, editors. Proceedings of the 2nd International Conference on Principles of Knowledge Representation and Reasoning (KR'91); 1991 Apr; Cambridge, MA, USA. San Mateo (CA): Morgan Kaufmann; 1991. p. 473–484.
4. Mascardi V, Demergasso D, Ancona D. Languages for programming BDI-style agents: an overview. In: Corradini F, Paoli FD, Merelli E, Omicini A, editors. WOA 2005: Dagli oggetti agli agenti. 6th AIIA/TABOO Joint Workshop "From Objects to Agents": Simulation and Formal Analysis of Complex Systems*; 2005 Nov 14–16; Camerino (MC), Italy. Bologna: Pitagora Editrice; 2005. p. 9–15.
5. Dennis LA, Farwer B. Gwendolen: a BDI language for verifiable agents. In: Löwe B, editor. Proceedings of the AISB 2008 Symposium on Logic and the Simulation of Interaction and Reasoning; 2008; Aberdeen, UK. Aberdeen: Society for the Study of Artificial Intelligence and Simulation of Behaviour (AISB); 2008. p. 16–23.
6. Bordini RH, Hübner JF, Wooldridge M. Programming Multi-agent Systems in AgentSpeak Using Jason. Chichester: John Wiley & Sons; 2007.
7. Collier RW, Russell S, Lillis D. Exploring AOP from an OOP perspective. In: Proceedings of the 5th International Workshop on Programming Based on Actors, Agents, and Decentralized Control (AGERE! 2015); 2015 Oct; Pittsburgh, PA, USA. New York (NY): Association for Computing Machinery; 2015. p. 25–36. doi:10.1145/2824815.2824818.
8. Braubach L, Pokahr A, Lamersdorf W. Jadex: a BDI-agent system combining middleware and reasoning. In: Unland R, Calisti M, Klusch M, editors. Software agent-based applications, platforms and development kits. Basel: Birkhäuser Verlag; 2005. p. 143–168. doi:10.1007/3-7643-7348-2_7.
9. Nakamoto S. Bitcoin: a peer-to-peer electronic cash system. [Online]. Bitcoin. 2008. Available from: https://bitcoin.org/bitcoin.pdf
10. Buterin V. (2013). Ethereum: A next-generation smart contract and decentralized application platform [online]. Ethereum. Available from: https://ethereum.org/en/whitepaper/
11. Tian F. A supply chain traceability system for food safety based on HACCP, blockchain & Internet of Things. In: Proceedings of the 2017 International Conference on Service Systems and Service Management (ICSSSM); 2017 Jun 16–18; Dalian, China. 2017. p. 1–6. doi:10.1109/ICSSSM.2017.7996119.
12. Calvaresi D, Dubovitskaya A, Calbimonte JP, Taveter K, Schumacher M. Multi-agent systems and blockchain: results from a systematic literature review. In: Demazeau Y, An B, Bajo J, Fernández-Caballero A, editors. Advances in Practical Applications of Agents, Multi-Agent Systems, and Complexity: the PAAMS Collection. Cham: Springer; 2018. p. 110–126. doi:10.1007/978-3-319-94580-4_9.
13. Ciatto G, Mariani S, Omicini A, Zambonelli F. From agents to blockchain: stairway to integration. Appl Sci. 2020;10:7460. doi:10.3390/app10217460.

