2024 IEEE 33rd International Symposium on Industrial Electronics (ISIE) | 979-8-3503-9408-5/24/$31.00 ©2024 IEEE | DOI: 10.1109/ISIE54533.2024.10595680

Enhancing Traceability in Flexible Production System: A Blockchain-Powered Approach in IEC 61499 Multi-Agent Control System

Midhun Xavier*, Sandeep Patil*, Valeriy Vyatkin* ‡
* Department of Computer Science, Computer and Space Engineering, Luleå Tekniska Universitet, Sweden
‡Department of Electrical Engineering and Automation, Aalto University, Espoo, Finland

Email: midhun.xavier@ltu.se, sandeep.patil@ltu.se, vyatkin@ieee.org

***Abstract*—This paper presents a novel approach for tracking products and processes within industrial multi-agent control systems by leveraging blockchain technology. The suggested solution facilitates the recording and validation of every step involved in creating a tailored product through the utilization of Ethereum-based smart contracts. The OWL ontology is used to describe agents and their capabilities and these software agents interact with IEC 61499 function blocks for process execution. The software agents record process events at each stage on the blockchain and the latter smart contract helps to trace and verify these process sequences of the customised product.**

***Index Terms*—Blockchain, Ethereum, IEC 61499, Multi-Agent System, OWL, Smart Contracts**

I. Introduction

In the realm of automation, achieving flexibility and reconfigurability is paramount for adapting to dynamic production requirements and evolving industrial landscapes. The utilization of Multi-Agent Systems presents a plethora of opportunities for industrial automation systems owing to their adaptable and reconfigurable structure [1]. Through the coordination and collaboration of autonomous agents, these systems facilitate agile responses to changes in operational needs, enabling seamless adjustments in manufacturing processes. These agents exhibit considerable advancements within Industry 4.0 initiatives by effectively addressing intricate challenges inherent in industrial control systems [2].

Agents utilize Semantic Web technologies [3] by annotating their actions and data with semantic metadata, enabling richer representation and understanding. With Semantic Web reasoning engines, agents can infer new knowledge, make informed decisions, and adapt their behavior dynamically [4]. Agents leverage semantic search techniques for precise information retrieval, achieve interoperability through adherence to common standards, and integrate heterogeneous data sources seamlessly, leading to more intelligent and effective interactions in Multi-Agent Systems. In this paper, W3C web ontology language (OWL) [5] is used to describe the capabilities and skills of the agent. The Multi-Agent Systems enhanced with the Semantic web ontology stack, provide scalable and flexible solutions for reconfigurable production lines to produce customised products on customer demand.

The IEC 61499 standard [6] is a framework for designing distributed control systems, particularly useful in industrial automation. It facilitates the development of Multi-Agent Systems by providing guidelines for modelling autonomous agents and coordinating their interactions in decentralized environments. Compared to the IEC 61131 standard, IEC 61499 is better suited for Multi-Agent Systems due to its focus on modularity, event-driven communication, and support for dynamic reconfiguration, which collectively enable greater flexibility and adaptability in response to changing operational requirements [7]. These features of IEC 61499 contribute [8] to achieving reconfigurability and flexibility by allowing agents to be easily added, removed, or modified without disrupting overall system functionality, thereby enhancing efficiency and responsiveness in industrial automation applications.

In the context of flexible and reconfigurable automation in Industry 4.0, ensuring traceability of products is a critical challenge. Ivan Jovović et.al [9] explore how the challenges of Industry 4.0 can be addressed by using emerging technologies such as 5G and Blockchain to improve the transparency, integrity and security of data. Blockchain technology offers a solution by providing a decentralized and immutable ledger for securely recording each step of the production process. By leveraging blockchain, stakeholders can transparently trace the lifecycle of products, from raw materials to finished goods, ensuring authenticity and accountability throughout the manufacturing process. Smart contracts [10] can automate and enforce predefined rules, further enhancing efficiency and reliability in product traceability within flexible and reconfigurable automation systems.

This paper presents the development of a Multi-Agent System leveraging the IEC 61499 standard to enhance flexibility and reconfigurability. We propose a solution empowered by smart contracts to address verification and validation challenges within Industry 4.0, harnessing the capabilities of blockchain technology. This solution records process sequences on the blockchain, enabling verification of product adherence to correct process orders. The immutable nature of blockchain records allows for real-time identification and removal of defective products while also providing customers

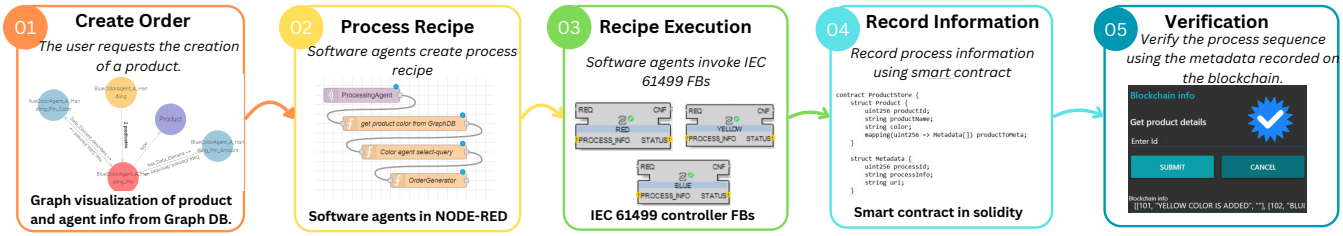

Fig. 1. Workflow for customised Product Process Verification.

with transparency through access to production process information via block verification.

The paper is organised as follows. Section II provides an overview of the advantages and challenges associated with integrating blockchain technology into industrial control systems. Section III provides an overview of Ethereum blockchain, InterPlanetary File System (IPFS), and Graph Databases, highlighting their roles and benefits in facilitating decentralized processing, secure data storage, and efficient modelling of complex relationships within the Multi-Agent System. Section IV explains in detail the methodology for the design and verification of Multi-Agent Systems using the concept of smart contracts. Section V describes the illustrative example of a developed Multi-Agent System and presents the result of the work. Finally, Section VI concludes the article and outlines future goals.

II. Blockchain in Industrial Control Systems

Research in the field of blockchain technology applications in industrial control systems (ICS) has been increasingly gaining attention in recent years [11], [12]. Blockchain can be used to record and verify the execution of processes in ICS, ensuring that the system is operating correctly and securely. This can be achieved by using smart contracts, which are self-executing contracts with the terms of the agreement directly written into code. Ethereum [13] is a decentralized blockchain platform featuring smart contract functionality, allowing for the creation of self-executing contracts with predefined rules and conditions. In industrial control systems, Ethereum and smart contracts can play a crucial role in building, verifying and validating process executions. Smart contracts automate predefined processes, ensuring consistency and reliability, while Ethereum's blockchain provides transparency and immutability, enabling stakeholders to verify and trace process executions. These smart contracts incorporate validation mechanisms to ensure adherence to criteria or standards, enhancing reliability and compliance. Overall, Ethereum and smart contracts offer a decentralized and transparent framework for efficient, reliable, and auditable process execution in industrial control systems.

One of the main applications of blockchain technology in ICS is data security. A study [11] presents a novel architecture, ICS-BlockOpS, that aims to improve industrial control system (ICS) data security by using blockchain technology. ICS-BlockOpS is designed to enhance data security by using an integrity-checking mechanism and data redundancy by applying an efficient replication mechanism. The prototype implementation of ICS-BlockOpS uses the Ethereum blockchain in a local network as part of the tamper-proofing mechanism and was tested in an operational six-stage water treatment plant. Another study by Tsai *et al.* [12] proposes a blockchain-based network architecture to secure the integrity of data in Industrial Control Systems (ICS) in Industry 4.0. The proposed architecture is implemented on physical industrial equipment, and it verifies and records the transmission of industrial control signals based on authority through a specialised client-server network model. The architecture aims to improve the security of ICS against adversarial attacks.

Previous research has explored blockchain in ICS for the security, transparency, and integrity of data [14], [15]. However, there has been limited research on its use in the verification and validation of execution processes in an industrial automation system. This paper addresses the traceability of products and processes from flexible and reconfigurable production systems by recording process sequences on the blockchain, facilitating real-time verification of product adherence to correct process orders. The immutable nature of blockchain records enables the prompt identification and removal of defective products, while also providing customers with transparent access to production process information through block verification.

III. Background

A. Ethereum

Ethereum [13] is a decentralized blockchain platform that enables the creation and execution of smart contracts and decentralized applications (DApps). Smart contracts are self-executing contracts written in code that run on the Ethereum blockchain. The Ethereum Virtual Machine is the runtime environment within Ethereum where smart contracts are executed, allowing for the decentralized processing of code and transactions on the network.

1) Smart contract: A smart contract [10] is a self-executing contract with the terms of the agreement directly written into code. It runs on a blockchain platform, such as Ethereum, and automatically executes actions when predefined conditions are met. In recording process data, smart contracts provide a secure and transparent way to store data on the blockchain. By defining functions within the smart contract code to input and store process data, it ensures immutability and tamper-proof recording of events. In a flexible production system, smart

contracts enhance traceability by creating an auditable record of each step in the production process. With data securely recorded on the blockchain, stakeholders can trace the origin and journey of products and verify authenticity. This increased transparency and traceability promote trust among participants and enable faster identification and resolution of issues in the production process.

2) Ethereum Virtual Machine (EVM): The Ethereum Virtual Machine [16] serves as the runtime environment for executing smart contracts on the Ethereum blockchain. Smart contracts, written in code, are automatically executed by the EVM when triggered by predefined conditions. For recording process data, developers write smart contracts with functions to input and store data securely on the blockchain, ensuring immutability and transparency. Users can interact with these contracts to input data, while querying functions allow access to recorded process data, facilitating a decentralised and auditable record of events.

B. InterPlanetary File System (IPFS)

IPFS, or InterPlanetary File System, is a decentralized protocol specifically designed to facilitate the storage and distribution of substantial data across a distributed network [17]. Utilizing IPFS for off-chain storage of large files, such as video or image records of product executions, this system avoids direct blockchain storage by generating a unique Content ID (CID) for each file and placing the actual data on a network of distributed nodes. This strategy alleviates the burden on individual blockchain nodes by offloading large data storage to the IPFS network, thus enhancing scalability and performance. The use of CIDs in blockchain transactions or smart contracts ensures the integrity and immutability of the data, making it resistant to tampering and alterations. In essence, IPFS offers a decentralized, efficient method for managing large data files, while still harnessing the security and immutability features of blockchain technology.

C. Graph Database (Graph DB)

A Graph Database utilizes graph structures—nodes, edges, and properties—to store data, offering significant advantages in environments like Multi-Agent Systems where complex relationships between entities like agents, skills, and product specifications need efficient modeling and querying [18]. This database format excels in navigation and data retrieval through powerful traversal capabilities, crucial for analyzing and optimizing agent interactions. Compared to traditional relational databases, it enhances scalability, flexibility, and performance, especially in dynamic environments.

IV. Methodology

A. Workflow Description

The workflow diagram of the verification of customised product process recipe is shown in Figure. 1. Initially, the user initiates the creation of a product by executing a transaction on the blockchain. Subsequently, software agents accept the order request and generate a process recipe tailored to the user's specifications. The agents then utilize IEC 61499 function blocks (FBs) to execute tasks according to the process recipe. During each step of the process, metadata including process description, status, timestamp, and additional information, such as videos or images are recorded using smart contracts. Finally, the verification of process sequences within the customised product's recipe is conducted by cross-referencing the metadata associated with the product on the blockchain.

B. Proposed Solution

The proposed architectural design for blockchain-powered product traceability in Multi-Agent System shown in Figure. 2. It is constructed employing various types of software agents, namely User agents, Processing agents, Execution agents (referred to as colour agents within this scenario), and a Verifying agent. These software agents can be developed in any programming environment, but, in this scenario, they are implemented using NODE-RED. It helps software agents by connecting nodes to perform tasks such as integrating IoT devices, interacting with APIs, and automating workflows. Its intuitive interface and extensive library of nodes make it easy to design complex logic without extensive coding, making it popular for IoT applications and automation tasks. This Multi-Agent System developed helps to make the production line flexible, configurable, and scalable.

Within this manufacturing marketplace, users have the ability to request customised products, with the generation process facilitated by software agents. When a user initiates an order, the user agent invokes a function within a smart contract to log the product's creation. This smart contract operates on an Ethereum Virtual Machine (Section III-A2), storing the product's ID and order specifications. The user agent logs the product details onto a Graph DB. Software agents analyse order requests and formulate process recipes. A processing agent generates proposals based on these recipes and distributes them to executing software agents, termed Colour Agents in this context. Executing agents assess the proposals, accepting or rejecting them based on their individual capabilities and constraints. The processing agent then evaluates the accepted proposals to identify the most optimal option, typically determined by the lowest cost of execution. These executing agents interact directly with IEC 61499 function blocks to execute skills at a low level, with execution agents utilizing MQTT communication protocol to execute their skills. Upon executing each process according to the recipe, the product metadata is stored via a smart contract call. Subsequently, the Verifying agent compares these metadata with the process recipe for validation purposes.

V. Case Study

A. User agent

When users request customised product colours along with the desired quantities, the user agent manages the order process. Upon receiving an order, the user agent assesses its compatibility with the agent's capabilities and skills stored in the Graph DB (Section III-C,) subsequently accepting or

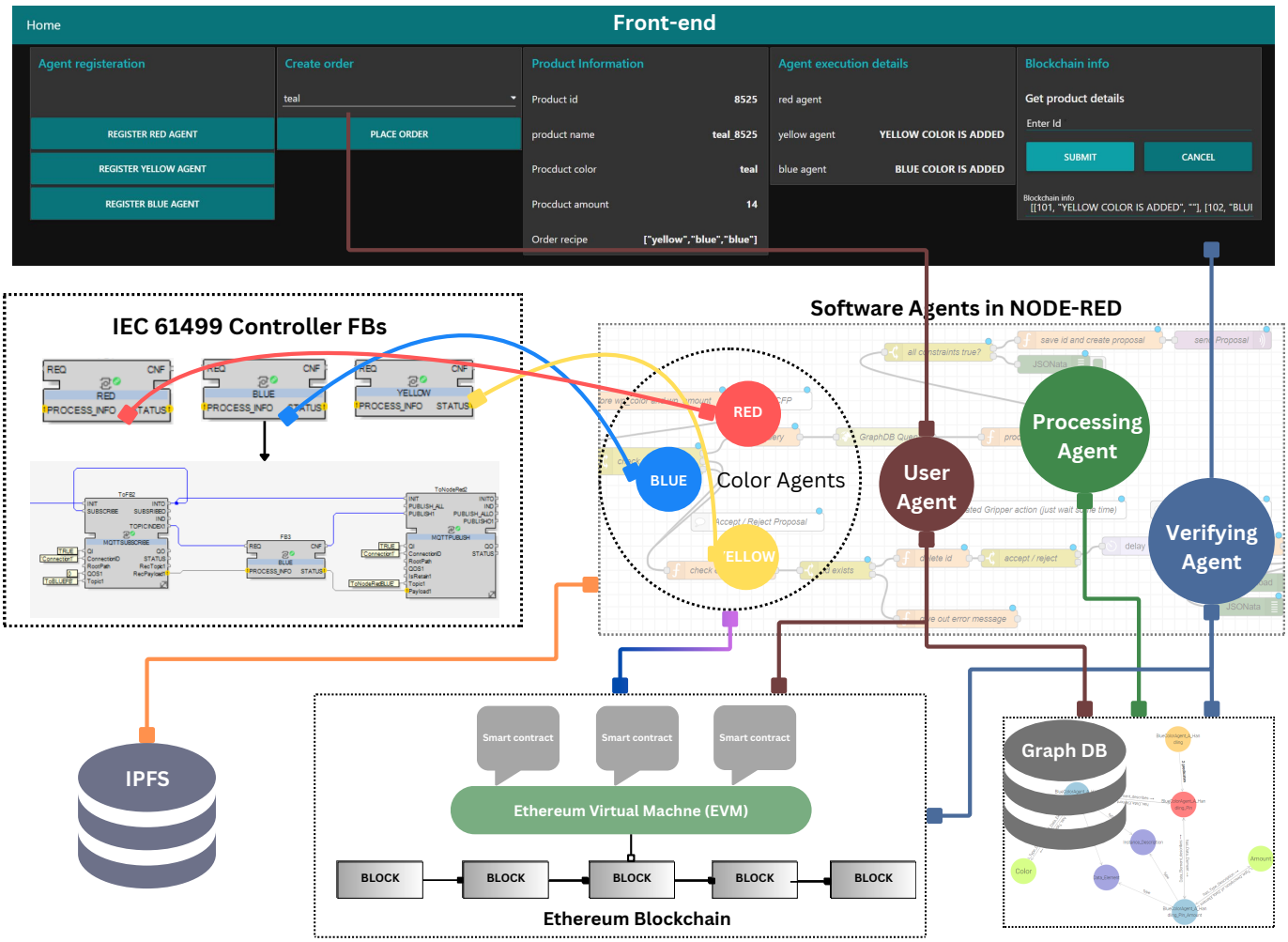

Fig. 2. Fostering Transparency: Architectural Design for Blockchain-Powered Product Traceability in Multi-Agent System

rejecting the order. Accepted orders are logged in the Graph DB using SPARQL queries, updating details such as product ID, name, colour, and quantity.

```
INSERT DATA {
agents:Product_${id} a agents:Product;
agents:hasProductId "${id}";
agents:hasName "${name}";
agents:hascolour "${colour}";
agents:hasAmount "${amount}".
}
```

The SPARQL query provided is designed to insert data about a product into a knowledge graph. The query specifies that the product is an instance of the ‘Product‘ class, and it associates the product with its ID, name, colour, and amount properties using the appropriate predicates (’agents:hasProductId’, ’agents:hasName’, ’agents:hascolour’, ’agents:hasAmount’). This allows for a structured representation of product information within the knowledge graph, facilitating efficient retrieval and querying of product data.

```
function mintProduct(uint256 _id, string memory
    _name, string memory _colour)
    public
    returns (uint256 productId)
{
    productId = _id;
    Product storage newProduct = idToProduct[
    productId];
    newProduct.productName = _name;
    newProduct.colour = _colour;
}
```

The provided Solidity code snippet defines a function named ‘mintProduct‘ which is responsible for creating a new product on the Ethereum blockchain. It takes parameters for the product’s ID, name, and colour and returns the ID of the newly minted product. Within the function, the ID is assigned to the ’productId’ variable, and the product’s name and colour are stored in a ‘Product‘ struct using the provided ID as the key in a mapping called ’idToProduct’.

B. Processing agent

The processing agent is responsible for crafting a process recipe for the user-requested product and subsequently overseeing its execution. After analyzing the process recipe, the processing agent formulates proposals and shares them with registered agents possessing equivalent capabilities. Execution

agents review the proposals and, if they meet all criteria, accept them along with pricing details, sending them back to the processing agent. After collecting accepted proposals, the processing agent identifies the optimal proposal based on cost efficiency and grants permission to an execution agent to execute the skill.

C. Execution Agents (colour agents)

Colour software agents manage the colour generation process, using primary colours Red, Yellow, and Blue to produce secondary and tertiary colours. Secondary colours, such as Green, Orange, and Purple, result from combining two primary colours, while tertiary colours, such as Teal, Chartreuse, Vermilion, Magenta, Violet, and Amber, stem from mixing primary and secondary hues. With primary colour agents Painting the production of various colours, user requests for specific colour products prompt collaboration among these agents. The system may accommodate multiple primary colour agents of the same type, each equipped with 'colour' and 'amount' data elements specifying the colours processable and the quantity of products that can be generated.

In this scenario, three distinct types of colour agents, namely red, yellow, and blue, are employed. The system allows agents to be registered and unregistered dynamically, enhancing flexibility and configurabilty. For this case study, the Ontology schema [3] is utilised to store agent and skill information in a Graph DB. Agents are registered through the execution of SPARQL queries, with each agent's details being stored in the GraphDB. An excerpt of a SPARQL query snippet illustrating the registration process of a yellow colour agent is presented below.

```
INSERT DATA {
agents:YellowcolourAgent_A rdf:type agents:
    colourAgent ;
agents:hasCapability agents:
    YellowcolourAgent_A_Painting.
agents:YellowcolourAgent_A_Painting VDI3682:hasInput
     agents:YellowcolourAgent_A_Painting_PIn;
VDI3682:hasOutput agents:
    YellowcolourAgent_A_Painting_POut;
  rdf:type agents:Painting;
agents:hasUrl "/YellowcolourAgent_A".
agents:YellowcolourAgent_A_Painting_PIn rdf:type
    agents:Product ;
DINEN61360:has_Data_Element agents:
    YellowcolourAgent_A_Painting_PIn_Amount ,
agents:YellowcolourAgent_A_Painting_PIn_colour.
agents:YellowcolourAgent_A_Painting_PIn_Amount rdf:
    type DINEN61360:Data_Element,
DINEN61360:Instance_Description ;
DINEN61360:has_Instance_Description agents:
    YellowcolourAgent_A_Painting_PIn_Amount ;
DINEN61360:has_Type_Description agents:Amount ;
DINEN61360:Expression_Goal "RequirementInteger" ;
DINEN61360:Logic_Interpretation "<=" ;
DINEN61360:Value 50 .
agents:YellowcolourAgent_A_Painting_PIn_colour rdf:
    type DINEN61360:Data_Element,
DINEN61360:Instance_Description ;
DINEN61360:has_Instance_Description agents:
    YellowcolourAgent_A_Painting_PIn_colour ;
DINEN61360:has_Type_Description agents:colour ;
DINEN61360:Expression_Goal "RequirementString" ;
DINEN61360:Logic_Interpretation "==" ;
DINEN61360:Value "Yellow" .
.....................................................
}
```

The given SPARQL query adds details about a yellow colour agent to a database, specifying its capability to perform painting tasks. It includes information about the input and output products involved in the painting process, such as the amount and colour of paint used and produced. Essentially, the query registers the yellow colour agent in the system and outlines its role in painting tasks with specific data attributes.

The colour agent receives proposals from the processing agent, which are then analysed to ensure that all constraints of the agent are met before acceptance. If a proposal satisfies these conditions, the colour agent accepts it and provides details regarding the execution price for the task. Once approval is granted by the processing agent, the colour agent proceeds to execute the skill, such as painting the product with the specified colour. These colour agents utilise IEC 61499 function blocks to execute the skill. Following the execution of each skill, the process description, process status, timestamp, and other information (i.e., video of execution or picture of the product) are added as metadata by invoking a smart contract function on the blockchain.

```
function addMetaData(
    uint256 _productId,
    uint256 _processId,
    string memory _processInfo,
    string memory _uri
) public {
    idToProduct[_productId].productToMeta[_productId
    ].push(
        Metadata(_processId, _processInfo, _uri)
    );
}
```

This smart contract function, named 'addMetaData', allows for the addition of metadata related to a product's process. It takes parameters including the product ID, process ID, process information, and a URI. Due to the challenge of storing large amounts of data directly on the blockchain, IPFS is utilised, as described above III-B. When extensive data such as videos of executions or images of products need to be stored, IPFS is leveraged, which generates a Content ID (CID). This CID can be encrypted, and the encrypted CID is then stored as a URI during the execution of smart contract calls.

D. Verifying agent

Once all process tasks within a recipe are executed by the colour agents, the verifying agent verifies whether the actual process recipe has been followed. The verifying agent retrieves the metadata by initiating the following smart contract call.

```
function retrieveProductMetaById(uint256 _id)
    public
    view
    returns (Metadata[] memory)
{
    Metadata[] memory derivedMeta = idToProduct[_id
    ].productToMeta[_id];
    return (derivedMeta);
}
```

Product metadata recorded can be obtained through the smart contract function 'retrieveProductMetaById'. The product metadata consists of the process description, the status of the process, the timestamp, and other information such as the execution video or the picture of the product. Eventually, this information is compared with the actual process recipe to see if any deviation has occurred.

Using the verifying agent, two issues were identified in the developed Multi-Agent System. The first issue arose when the processing agent granted permission to execution agents to execute skills according to the process recipe, but these executions were carried out in a different order than specified. This was resolved by limiting the processing agent to execute one task at a time and allowing the execution agent to acknowledge back to the processing agent. The second issue was more complex as it was discovered that execution agents sometimes initiated the execution of a skill but failed to complete it due to various issues. The processing agent expected an acknowledgement from the execution agent, but it was not received. To address this, the processing agent identified another agent to execute the same skill. The verifying agent retrieved metadata and identified an incomplete execution status, which led another agent to execute the skill again, resulting in defective products. This functionality improves trust and reliability, reducing the risk of defective products entering the market. Manual inspection of recorded videos or images during skill execution can help identify issues and improve the Multi-Agent System's performance.

VI. Conclusion and future work

This paper presents a novel approach for tracking products and processes within industrial multi-agent control systems by leveraging blockchain technology. The proposed solution facilitates the recording and validation of every step involved in creating a customised product through the utilisation of Ethereum-based smart contracts. By employing the OWL ontology to describe agents and their capabilities, and integrating IEC 61499 function blocks for process execution, the system enables flexible and reconfigurable production lines to produce customised products on demand. Through the implementation of smart contracts on the Ethereum blockchain, process events are recorded and traced, ensuring the adherence of customised products to correct process sequences. The user can also retrieve the same information about the product from the blockchain, which makes the system more transparent and reliable.

In future iterations, Non-Fungible Tokens (NFTs) could be integrated to uniquely represent each customised product, providing a digital certificate of authenticity containing metadata about its origin and manufacturing process. These NFTs could also enforce royalty agreements for creators and manufacturers, track product customisation attributes, and ensure transparency throughout the supply chain, enhancing trust and enabling personalized product experiences for customers. By leveraging NFT technology, the system can offer enhanced traceability, authenticity, and value capture for stakeholders involved in the production and distribution of customised products.

VII. Acknowledgements

This research was supported, in part, by the Horizon Europe Zero-SWARM project funded by the European Commission (grant agreement: 101057083).

References

[1] A. Nilsson, F. Danielsson, and B. Svensson, "Customization and flexible manufacturing capacity using a graphical method applied on a configurable multi-agent system," *Robotics and Computer-Integrated Manufacturing*, vol. 79, p. 102450, 2023. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0736584522001326

[2] S. Karnouskos, P. Leitao, L. Ribeiro, and A. W. Colombo, "Industrial agents as a key enabler for realizing industrial cyber-physical systems: Multiagent systems entering industry 4.0," *IEEE Industrial Electronics Magazine*, vol. 14, no. 3, pp. 18–32, 2020.

[3] A. Köcher, C. Hildebrandt, L. M. Vieira da Silva, and A. Fay, "A formal capability and skill model for use in plug and produce scenarios," in *2020 25th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA)*, vol. 1, 2020, pp. 1663–1670.

[4] W. Ochoa, F. Larrinaga, and A. Pérez, "Context-aware workflow management for smart manufacturing: A literature review of semantic web-based approaches," *Future Generation Computer Systems*, 2023.

[5] M. Hadzic, E. Chang, P. Wongthongtham, and T. Dillon, *Ontology-based multi-agent systems*. Springer, 2009.

[6] "IEC 61499-1: Function Blocks Part 1: Architecture," 2012.

[7] C. Mulubika and A. Basson, "Comparison of iec 61499 and agent based control for a reconfigurable manufacturing subsystem," in *Proceedings of the in COMA13, International Conference on Competitive Manufacturing*, 2013, pp. 283–288.

[8] G. Lyu and R. W. Brennan, "Multi-agent modelling of cyber-physical systems for iec 61499-based distributed intelligent automation," *International Journal of Computer Integrated Manufacturing*, pp. 1–27, 2023.

[9] I. Jovović, S. Husnjak, I. Forenbacher, and S. Maček, "Innovative application of 5g and blockchain technology in industry 4.0," *EAI Endorsed Transactions on Industrial Networks and Intelligent Systems*, vol. 6, no. 18, 2019.

[10] G. A. Oliva, A. E. Hassan, and Z. M. Jiang, "An exploratory study of smart contracts in the ethereum blockchain platform," *Empirical Software Engineering*, vol. 25, pp. 1864–1904, 2020.

[11] A. Maw, S. Adepu, and A. Mathur, "Ics-blockops: Blockchain for operational data security in industrial control system," *Pervasive and Mobile Computing*, vol. 59, p. 101048, 2019.

[12] Y. C. Tsai, I.-H. Liu, and J. S. Li, "Blockchain-based verification mechanism for industrial control system," in *27th International Conference on Artificial Life and Robotics, ICAROB 2022*. ALife Robotics Corporation Ltd, 2022, pp. 309–312.

[13] V. Buterin, "A next-generation smart contract and decentralized application platform," 2014. [Online]. Available: https://github.com/ethereum/wiki/wiki/White-Paper

[14] S. Schorradt, E. Bajramovic, and F. Freiling, "On the feasibility of secure logging for industrial control systems using blockchain," in *Proceedings of the Third Central European Cybersecurity Conference*, 2019, pp. 1–6.

[15] J. Stodt, D. Schönle, C. Reich, F. Ghovanlooy Ghajar, D. Welte, and A. Sikora, "Security audit of a blockchain-based industrial application platform," *Algorithms*, vol. 14, no. 4, p. 121, 2021.

[16] E. Hildenbrandt, M. Saxena, N. Rodrigues, X. Zhu, P. Daian, D. Guth, B. Moore, D. Park, Y. Zhang, A. Stefanescu *et al.*, "Kevm: A complete formal semantics of the ethereum virtual machine," in *2018 IEEE 31st Computer Security Foundations Symposium (CSF)*. IEEE, 2018, pp. 204–217.

[17] J. Benet, "Ipfs-content addressed, versioned, p2p file system," *arXiv preprint arXiv:1407.3561*, 2014.

[18] Ontotext, "Graphdb documentation," https://graphdb.ontotext.com/documentation/10.0/references.html, accessed: February 28, 2024.