Resource-Based Blockchain Integration in Agri-food Supply Chains

Devis Bianchini, Valeria De Antonellis, Massimiliano Garda, and Michele Melchiori(✉)

Department of Information Engineering, University of Brescia, Via Branze 38, 25123 Brescia, Italy
{devis.bianchini,valeria.deantonellis,massimiliano.garda, michele.melchiori}@unibs.it

Abstract. In today's agri-food 4.0 supply chains, food traceability is crucial for enhancing transparency, safeguarding consumer health, and fostering trust in the food ecosystem. Blockchain Technology (BCT) is proposed as a solution, offering immutable, transparent, and decentralized data persistence capabilities. The complexity increases in scenarios with intertwined supply chains, where actors engage with various BCTs, each having its own technological infrastructure, programming language for Smart Contracts, and strategies for handling large volumes of on-chain data storage, to save costs and ensure scalability. In this paper, we propose to treat traceability data exchanged among supply chain actors as resources, enabling standardized interactions with BCTs, rather than focusing on specific storage technologies. Our approach: (i) leverages a base model to create tailored views for modeling data resources in traceability for the agri-food supply chain, aligning data persistence with the Access Level requirements of the supply chain actors; (ii) defines a Resource-Oriented approach for integrating BCTs in intertwined supply chains, guided by these views, in order to develop BCT-based Web Applications. Experiments with a real case study demonstrate the benefits of our approach in addressing BCTs integration issues. In particular, in this paper we focus on how this approach supports enhancing agri-food 4.0 supply chains consumers trust and transparency.

Keywords: Blockchain Integration · Blockchain in Supply Chains · Web Applications · Smart Contracts · Resource-Oriented Architecture

1 Introduction

In recent years, blockchain technology (BCT) has been widely considered for food supply chains to address the increasing demands for transparency, security, and traceability, thereby protecting consumers' health and enhancing trust in the entire food ecosystem [11]. BCT can help establish "credibility attributes" for consumers, such as preventing fraud, counterfeiting, the use of harmful treatment products, and the presence of contaminants. However, the complexity rises as actors may participate in intertwined supply chains, each utilizing its own BCT [12]. Consequently, these actors need to engage with various technological infrastructures and programming languages for Smart Contracts, along with different data storage solutions to minimize costs and tackle scalability challenges [21], especially when handling large volumes of data.

A data-centric strategy has facilitated the adoption of resource-oriented methodologies, employing the widely recognized REpresentational State Transfer (REST) architecture to model the interactions between actors and BCTs. By treating traceability data to be exchanged among actors as resources, rather than focusing on the specific storage technologies, it is possible to enable standardized interactions of actors with different BCTs through lightweight APIs. However, existing approaches [7,20] propose ad-hoc solutions based on a specific BCT, they do not specifically address food traceability using RESTful APIs and they do not delve into details concerning on-chain data storage costs. Requirements like trustworthiness, transparency, costs control, usage of the blockchain storage, privacy implementation have to be leveraged to decide whether to persist data on-chain or off-chain [3,6].

Motivating Scenario. Consider a scenario in the agri-food domain with three interconnected supply chains (Fig. 1), utilizing heterogeneous BCTs: (i) the *DOP*[1] *Wine Supply Chain*, which leverages the Ethereum BCT and involves a variety of actors (grape growers, wine producers, bulk distributors, transit cellars, fillers/packers, and wine retailers); (ii) the *DOCG Wine Supply Chain*, which uses the Hyperledger Fabric BCT; (iii) the *Balsamic Vinegar Supply Chain*, also employing the Ethereum BCT. Some actors may participate in multiple supply chains. For example, the bulk distributor, focused on a specific regional wine market, is involved in the *DOP Wine Supply Chain* but may also collaborate with other wine producers in different categories, such as in the *DOCG Wine Supply Chain*. Moreover, the traceability process, and data supporting it, can serve different perspectives of involved actors. For example, product origin, identity and quality, as well as provenance and authenticity of purchased goods are of interest for consumers, while a food certification organisation can look for traceability data aimed at the certification purposes and only partially overlapping with data provided to consumers.

Contributions and Paper Organisation. The approach presented in this paper extends the preliminary effort made in [4], where we discussed about the challenges of blockchain integration in intertwined supply chains, presenting an agri-food motivating scenario. With respect to [4], this paper proposes an organized and step-by-step methodological approach to design a resource-oriented architecture for the integration of BCT-based applications in intertwined supply chains. In particular, we aim to demonstrate that the approach is generally scalable and cost-effective, while on the application side, we focus on how it can contribute to consumer trust and transparency. A base data model ensures

[1] The DOP and DOCG labels are designed to protect the quality of Italian wines, similar to their counterparts within the European Union.

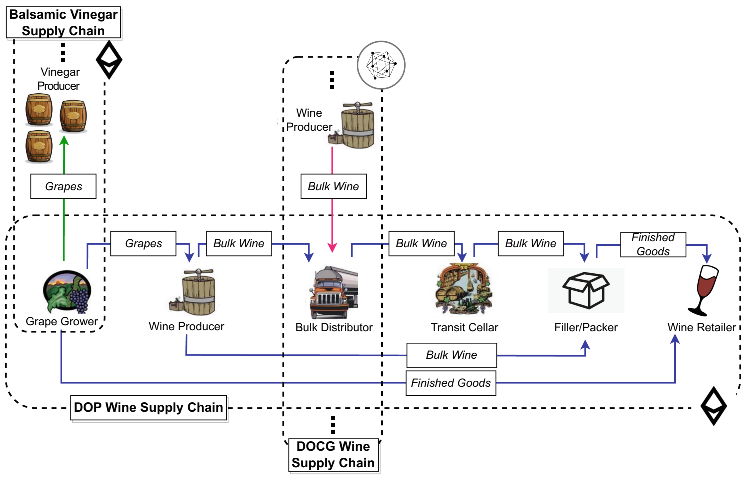


Fig. 1. Example of intertwined supply chains in agri-food (arrows indicate the flows of intermediate and final goods exchanged between actors).

consistent representation of wine product information across different blockchain technologies. Consumers benefit from a uniform way of accessing and interpreting product data, regardless of the underlying blockchain used by different Supply Chain actors. The approach is organised over: (i) BCT-independent steps, where resources within the supply chain and corresponding *RESTful Services* to share resources between actors are modelled, disregarding any specific BCT for persistence; (ii) BCT-dependent steps, where RESTful services are mapped to Smart Contracts of specific BCTs, handling on-chain/off-chain storage requirements to address cost containment. In this way, the complexity of various BCTs in developing BCT-based Web Applications is concealed behind REST APIs. We performed experiments regarding on-chain storage costs in the agri-food sector, as illustrated in Fig. 1, using two widely adopted BCTs. The results demonstrate the benefits of a resource-oriented approach compared to a direct use of Smart Contracts for integrating these BCTs. The paper outline is the following: in Sect. 2, related work are discussed; in Sect. 3 the methodological approach is detailed; implementation and experimental evaluation are described in Sect. 4; a discussion on the broader implications of adopting this approach from a consumer perspective is given in Sect. 5, finally, Sect. 6 closes the paper.

2 Related Work

In recent years, the proliferation of different BCTs has led to two main research areas: (i) *blockchain integration*, focusing on integration of BCT with existing software systems at the application level; and (ii) *blockchain interoperability*, aimed at enabling the transfer/exchange of data between technologically distinct

BCTs. This paper contributes to the second domain, by introducing a resource-oriented approach to BCT integration in interconnected supply chains.

Resource-oriented approaches to BCT integration emphasize the information exchanged (e.g., products and documents in a supply chain) rather than on the interactions between actors, which are more in the focus of WSDL-like service-oriented interfaces over BCTs [7,20]. Current solutions promote a lightweight interface for managing read/write operations on specific BCTs. For example, the work in [18] establishes a REST API layer to invoke the chaincode of a Hyperledger Fabric implementation within a coffee supply chain. In [9], a permissioned blockchain is utilized in an agricultural supply chain, employing REST APIs for interaction with the BCT regarding registration, transaction management, and querying. The work [10] discusses a use case in a textile supply chain, illustrating the coexistence of BCT and Big Data. These proposals typically do not deal with on-chain data storage costs or provide a comprehensive methodology for blockchain integration, instead emphasizing privacy and authentication mechanisms and patterns.

Blockchain integration in agri-food supply chains mainly focuses on *food distribution*, *food origin and sourcing*, and *food safety and quality* [13,15,16]. For example, the TE-FOOD system combines public and private BCTs with data from IoT sensors, employing Smart Contracts to gather data at each stage of the farm-to-table pork supply chain. The Certified Origins EVOO Oils project utilizes Oracle's private BCT to improve transparency in the production chain of extra virgin olive oil. These strategies emphasize the specific, ad-hoc application of particular BCTs but do not explicitly address food traceability through RESTful API.

3 The Model-Based Integration Approach

In this section, we describe a model-based approach for designing resource-oriented architectures for blockchain integration. The approach consists of two distinct phases, each comprising a set of *steps*.

– *BCT-independent* steps (Sect. 3.1) are carried out irrespective of the target BCT. Specifically: (i) *resource modelling* utilizes a conceptual data model of the supply chain data, which will be stored on the blockchain as resources; (ii) *service modelling* focuses on creating Resource-Oriented Services, which are made accessible via REST APIs, while hiding the complexity of managing resource persistence.
– *BCT-dependent* steps (Sect. 3.2) are tightly-coupled with the target BCT. In particular: (i) *software components definition* involves the deployment of software components that manage the invocations of the REST API for publishing and reading resources, as well as executing any actions on those resources; (ii) *mapping of storage methods* implements the connection between software components and the blockchain (e.g., for accessing Smart Contract functionalities) as well as with other data persistence providers (e.g., decentralized external file systems).

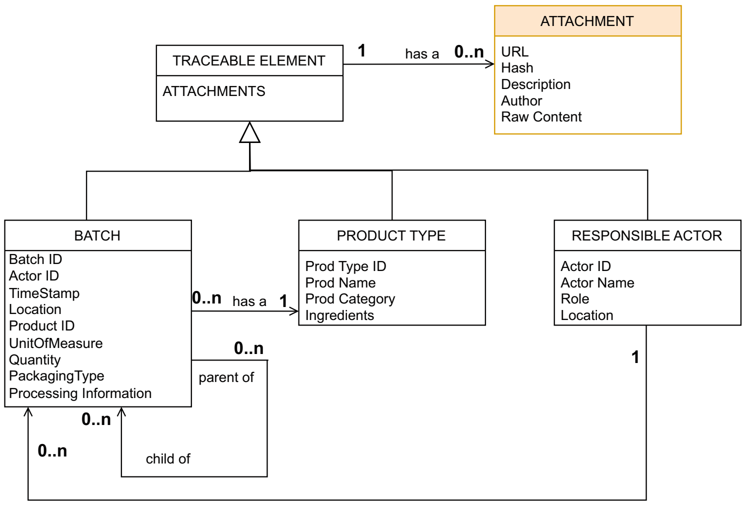


Fig. 2. Base data model for traceability in agri-food supply chains.

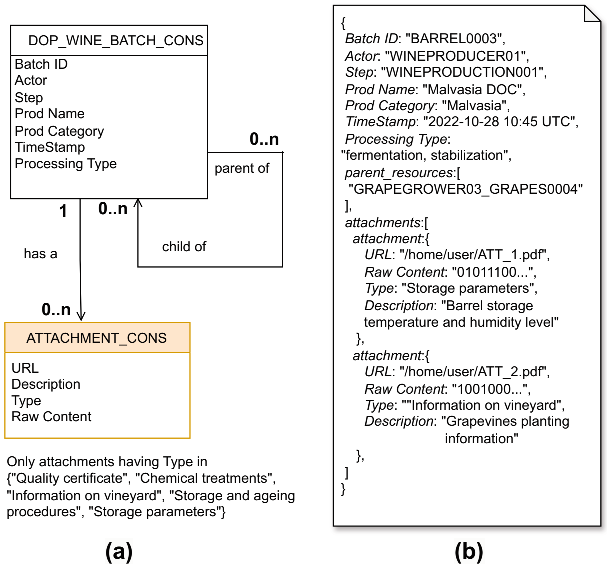
3.1 BCT-Independent Steps

In agri-food supply chains, which commonly operate according to a batch manufacturing model, traceability focuses on monitoring flows and transformations of product batches. These batches may consist of packaged products, traded items (such as cases or cartons), or logistic units (e.g., bins or containers). In this landscape, traceability can assume various perspectives. For example, product origin, identity and quality, as well as provenance and authenticity of purchased goods are of interest for consumers, while a food certification organisation searches for traceability data aimed at the certification purposes and only partially overlapping with data provided to consumers [17]. Each perspective leads to distinct *use cases*. In the scope of each use case, our approach aims at abstracting traceability data by means of *resources* upon which *RESTful services* are in charge of implementing read/write operations to/from the blockchain or other persistence providers.

1) Resource modelling. Resource modelling consists of three sub-steps, which are described in the following paragraphs.

1.1) Selection of a base data model. To showcase our approach, we present a base data model for describing traceability data within an agri-food supply chain, inspired by the analysis of well-known literature on agri-food traceability (e.g., [17]). Noteworthy, the approach described in this paper remains neutral with respect to the selection of the base data model, which can be chosen, for example, from the literature. We propose here a (simplified) example of base model that is depicted in Fig. 2 as UML class diagram. This model encapsulates the essential data for tracing batches in the agri-food sector regardless of the target use case.

Example. The model in Fig. 2 focuses on the concept of `TRACEABLE ELEMENT`, which is specialised into the concepts of `BATCH`, `PRODUCT TYPE`, `RESPONSIBLE ACTOR`, and that can have some `ATTACHMENT`s. The auto-association on `BATCH` permits to track relationships between batches of product at different transformation and distribution steps of the supply chain.



```
{
 Batch ID: "BARREL0003",
 Actor: "WINEPRODUCER01",
 Step: "WINEPRODUCTION001",
 Prod Name: "Malvasia DOC",
 Prod Category: "Malvasia",
 TimeStamp: "2022-10-28 10:45 UTC",
 Processing Type:
"fermentation, stabilization",
 parent_resources:[
  "GRAPEGROWER03_GRAPES0004"
 ],
 attachments:[
   attachment:{
      URL: "/home/user/ATT_1.pdf",
      Raw Content: "01011100...",
      Type: "Storage parameters",
      Description: "Barrel storage
      temperature and humidity level"
   },
   attachment:{
      URL: "/home/user/ATT_2.pdf",
      Raw Content: "1001000...",
      Type: ""Information on vineyard",
      Description: "Grapevines planting
                   information"
   },
 ]
}
```

Fig. 3. (a) Consumer-oriented View in the DOP Wine Supply Chain, and (b) resource representation for a batch (JSON notation).

1.2) Creation of views on the base data model. The base data model is leveraged to create customised views, modelling the resources for a specific use case. Each view, derived from the base model, contains: (i) relevant classes of resources, (ii) attributes and their possibly renaming, (iii) constraints, with respect to the base data model. Formally, this can be obtained by applying a combination of standard relational algebra operators, as well as by adding constraints, over the base model. Figure 3(a) illustrates a view obtained from the base model for a consumer-oriented perspective over traceability data, where the information to be exposed to final consumers essentially regards the timeline of production and distribution steps, and information meant to certificate both product origin and quality. In the view, `DOP_WINE_BATCH_CONS` models a generic batch over the DOP Wine Supply Chain, whereas `ATTACHMENT_CONS` models complementary batch documentation, providing the consumer with origin and quality details (e.g., wine quality certificates, or descriptions of storage and ageing procedures of the batch). These classes represent *resources*, which serve as containers for traceabiity data that a supply chain actor intends to publish on the blockchain during the execution of a specific step. Creating views for specific use cases, leveraging the same base model, ensures inherent homogeneity of views, coherency across different views over the same supply chain and a faster modelling process. In addition, views can be easily maintained also in the case of evolution of the traceability data model (e.g., new classes or attributes).

Example. Figure 3(b) is an example of a JSON serialization of a resource from the DOP Wine Supply Chain, compliant with the view presented in Fig. 3(a). This resource contains information about a barrel (i.e., the considered batch) in the Wine Producer step, identified by the `WINEPRODUCER01_BARREL0003` ID. The properties of the resource include the product name (Malvasia DOC), product category (Malvasia), timestamp, and processing type (fermentation, stabilization). Additionally, the resource is further detailed through attachments (two PDF files). It is to noted that the resource with ID `GRAPEGROWER03_GRAPES0004` mentioned in the figure is the batch received as input by the Wine Producer (i.e., it is the parent resource) from the actor Grape Grower `GRAPEGROWER03` (who assigned the ID `GRAPES0004` to the product batch in its step). This parent resource is associated with the current resource for permitting traceability.

Table 1. Access level and large data specification for a step S of the Wine Supply Chain example.

Class (from V)	Attribute Set (D)	Access Level (AL)	Large Data (LD)
DOP_WINE_BATCH_CONS	Batch ID	SupplyChain	False
	Prod Name, Prod Category, TimeStamp, Location	All	False
	Processing Type	Private	False
ATTACHMENT_CONS	URL, Description, Type, Raw Content	All	True

1.3) Specification of additional user requirements on the views. Traceability of products involves recording and monitoring their origins, transitions, and destinations along the supply chain. This is accomplished by leveraging recorded identifiers for the batches throughout the supply chain [11]. In this respect, each actor involved in a step, whether it be food gathering, transportation, or distribution, is responsible for providing information (either manually or through sensors), regarding batches and processing details specific to the actor's steps. Actors define the desired visibility of their traceability data by specifying *access level* requirements. Additionally, they provide an indication of the *data size*. Access level specification, along with data size, are used to implement resource persistence both correctly and efficiently, and may suggest storing specific data either on-chain or off-chain [6], as discussed in Sect. 3.2.

Let V be a view, D a set of one or more attributes of a class in V and S a step of the supply chain. An *Access Level* specification (in brief, AL) defines who has access to an instance of D and is denoted as: $AL(V, D, S) \rightarrow \{All, Private, SupplyChain\}$, where All stands for public visibility, $Private$ for attributes accessible only by the responsible actor for S, and $SupplyChain$ to limit the visibility scope to the supply chain actors (excluding consumers). Instead, a *Large Data* specification (in brief, LD) is apt to specify whether the size of D is either large or not and is denoted as: $LD(V, D, S) \rightarrow \{True, False\}$, where $True$ means that instances of D include large data objects (e.g., documents, images).

Example. Table 1 presents the desired specifications for a step S of the DOP Wine Supply Chain. The choices of access levels are typically guided by desired requirements such as relevance, trustworthiness, and transparency of data.

2) Service modelling. Serialized resources can be submitted to (or retrieved from) the data persistence system by supply chain actors via their front-end applications, which trigger the invocation of what are known as Resource-Oriented Services (abbreviated as ROS), made available through a REST API (Table 2). Each ROS operates on a resource as defined earlier. A ROS is characterized by its URL, an HTTP verb (indicating the read/write operation to be conducted on the resource, such as GET or POST), and a binding that specifies how the resource should be serialized (e.g., in JSON format).

Table 2. Services of the DOP Wine Supply Chain REST API.

Service Description	HTTP Verb	End point Path
Register a new batch	POST	`/{step}/batches`
Retrieve a batch	GET	`/{step}/batches/{batch_id}`
Register a new parent batch	POST	`/{step}/batches/{batch_id}/parents`
Retrieve parent batches	GET	`/{step}/batches/{batch_id}/parents`
Retrieve a parent batch	GET	`/{step}/batches/{batch_id}/parents/{par_batch_id}`

Example. To publish the resource depicted in Fig. 3(b), the Wine Producer `WINEPRODUCER01` will use their front-end application to invoke the "Register a new batch" service as shown in Table 2. The resource's JSON serves as the payload for the HTTP request sent to the REST API, with the `step` parameter being replaced by `WINEPRODUCTION001` (i.e., the request will be `POST /WINEPRODUCTION001/batches`)

3.2 BCT-Dependent Steps

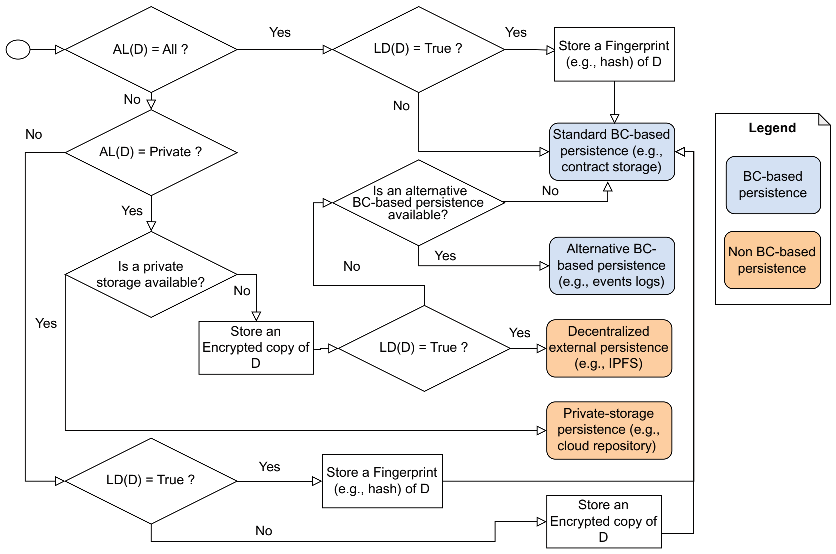
1) Software components definition. Service invocation requests issued to the REST API, as defined in the *service modelling* step, are dispatched to different typologies of software components providing implementation to ROS actions, namely *POST component* and *GET component*. In particular, requests for publishing resources on the storage system are routed to a *POST component*, whilst requests to read resources are routed to a *GET component*. These components deal with invoking lower level storage methods, performing ROS payload transformations and corollary operations (e.g., data encryption, checking integrity constraints). In order to store resources permanently, as the effect of POST requests, a general issue in the design of components is to choose a suitable storage mechanism based on the need of fulfilling the user's non-functional requirements on the views as discussed in Sect. 3.1. Figure 4 describes a decision chart to assist with the design aspects of persistence in POST components of a ROS. Based on a specification, like the one given in Table 1, the chart suggests among several possible types of storage strategies (coloured blocks) for data regarding an Attribute Set D, and produced in the context of a step S: (a) *non blockchain-based persistence*, to store the data into either a private or a decentralised storage, or (b) *blockchain-based persistence*, when the data (or a hash fingerprint of it, depending whether it is a large data item or not) is stored on-chain.

In particular, for large data (i.e., when the LD property is True) or private data, it is advisable to use non-blockchain-based persistence. This approach typically involves storing the data on decentralized storage systems like IPFS (InterPlanetary File System) and then publishing its hash on the blockchain to ensure data immutability. However, it is important to note that BCTs are generally not optimized for handling large data objects. For data that needs to be shared, blockchain-based persistence is advisable. Additionally, certain BCTs offer alternative storage mechanisms, such as the Ethereum event logs, which can exploited to reduce storage costs (please refer to Sect. 4 for examples).

2) Mapping of storage methods. In the context of a POST/GET request, depending on the required resource persistence, the associated POST/GET component utilizes: (i) methods of Smart Contracts, identified by their blockchain address, which are designed to store (or retrieve) data on the blockchain; (ii) methods of External File Storage, used to upload (in the case of POST) or retrieve (in the case of GET) raw files from a decentralized external storage system (e.g., IPFS). It is worth noting that the storage methods enabling interaction with BCTs and decentralized external file storage providers are typically incorporated into technology-specific libraries, which are utilized by the above mentioned software components.

4 Architecture for BCT Integration

The approach described in the previous section has been applied in a real-world scenario, involving two wine supply chains from the motivating example presented in Sect. 1: the *DOP Wine Supply Chain*, which leverages the Ethereum



AL(D) = All ?
Yes
No
LD(D) = True ?
Yes
No
Store a Fingerprint (e.g., hash) of D
Standard BC-based persistence (e.g., contract storage)
Legend
BC-based persistence
Non BC-based persistence
No
AL(D) = Private ?
Yes
Is an alternative BC-based persistence available?
No
Yes
Alternative BC-based persistence (e.g., events logs)
Is a private storage available?
No
Yes
No
Store an Encrypted copy of D
LD(D) = True ?
Yes
Decentralized external persistence (e.g., IPFS)
Private-storage persistence (e.g., cloud repository)
LD(D) = True ?
Yes
Store a Fingerprint (e.g., hash) of D
No
Store an Encrypted copy of D

Fig. 4. Decision support chart for resource persistence in the design of POST components.

BCT (abbreviated as ETH), and the *DOCG Wine Supply Chain*, which utilizes the Hyperledger Fabric BCT (abbreviated as HL Fabric). Figure 5 showcases the resource-oriented architecture, wherein two *integration layers* are responsible for handling read/write operations on resources from/to: (i) the two BCTs; (ii) other persistence providers, such as IPFS, which has been employed as an external decentralised storage system to store resources data off-chain. Software components enable the interaction with the respective BCTs by invoking Smart Contract methods, while interactions with the distributed external file storage are managed through IPFS storage methods. In Fig. 5, two different data flows have been highlighted, one regarding the registration of a batch on the DOP Wine Supply Chain, the other for registering a new batch on the DOCG Wine Supply Chain. Both requests are triggered by an actor, participating in both the supply chains, through the Front-end Web application of the actor, and dispatched by the REST APIs of the two supply chains to the respective POST software components for on-chain/off-chain resource data persistence. As shown in Fig. 5, adopting a resource-oriented architecture allows for hiding the heterogeneous technologies behind the REST APIs. The specification of the REST APIs (complying with the OpenAPI standard) along with the source code of the Web application (implemented in React.js) are available at [1].

Smart Contracts for On-Chain Resource Data Persistence. Considering the consumer-oriented view over traceability data for the DOP Wine Supply Chain (detailed in Sect. 3.1), to register resource data items on the Ethereum

BCT a unique Smart Contract has been generated[2] starting from the classes of the view in Fig. 3 (named `StepContract`, source code in Solidity programming language available at [1]). The `StepContract` contains methods to store on-chain batch information (`publishBatch` method) as well as IPFS-generated hashes of attachments (`storeHash` method). Noteworthy, the code of the `StepContract` has been deployed multiple times, one for each step of the DOP Wine Supply Chain. An analogous procedure has been conducted for generating the code of the Smart Contract for the HL Fabric BCT of the DOCG Wine Supply Chain (written in Go programming language and available at [1]).

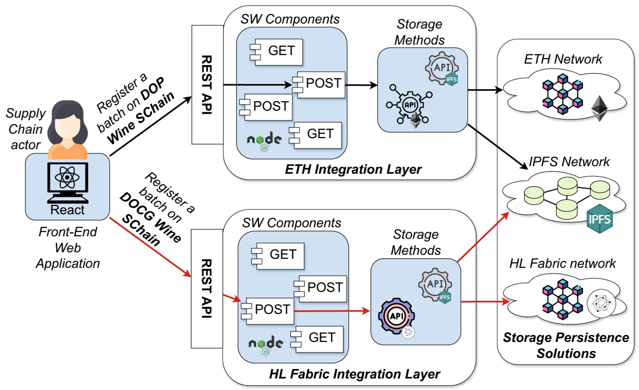


Fig. 5. REST-based architecture for two supply chains.

4.1 Experimental Evaluation

The primary focus of the experimental evaluation is the control and limitation of on-chain storage for the Ethereum BCT, as the so-called *gas mechanism* incurs significant economic costs for data storage transactions. In contrast, HL Fabric does not have a similar concept of storage cost; however, efficiency metrics (such as transaction rate, latency, and throughput) can be evaluated by creating benchmarks, as we outlined in [1]. In the Ethereum BCT, the costs associated with a Smart Contract primarily depend on the size of the deployed code (in bytes) and the actual invocations of Smart Contract methods (e.g., to modify the contract's state variables).

Regarding the first cost factor, resource modeling contributes to simplifying the business logic of Smart Contracts associated with Resource-Oriented Services. The second factor is impacted by the complexity of the resources, specifically: (a) the *primitive data types* and *data structures* used to store resource

[2] Using techniques for semi-automatic code generation from conceptual models (process not detailed here, as it is out of the scope of this paper).

data; and (b) the *type of memory* allocated for these data types and structures, which in the Ethereum BCT can encompass both *contract state storage* and *events/logs*. To provide the average price in Euros (€), the following reference values were applied (observed at the time of the tests): (a) a gas price of 25 Gwei (a sub-unit of Ether equal to 10^{-9} Ether); and (b) an Ether value of 1599.40 €. The gas price and the gas limit (i.e., the maximum amount of gas a user is willing to pay for a successful transaction) determine the total cost of the transaction.

Table 3. Cost comparison: (a) saving variables and IPFS Multihashes as strings (contract state storage), (b) saving variables as `bytes32` (using events and logs)

Variable type	Gas used (avg)	€ (avg)
Single string	112,506	4.50
Array of strings (1 string)	78,356	3.134
Entire structure of the batch	230,817	9.23
Single Multihash	114,007	4.56
Array of M.hashes (1 M.hash)	79,871	3.194
(a)		
Variable type	Gas used (avg)	€ (avg)
Single Multihash	29,425	1.18
Array of M.hash (1 M.hash)	10,056	0.402
Entire structure of the batch	30,257	1.21
(b)		

Contract State Storage and Logs. The tests outlined below aim to evaluate not only the type of memory (i.e., either contract storage or events/logs) but also the various data types and structures provided by Solidity for storing text data. This focus is important because, as shown in the example in Fig. 3(b), the resource information primarily has a textual representation. The first test was conducted using string-type variables stored in the contract state storage. Table 3(a) displays the costs associated with writing the so-called Multihash (32 bytes in length) as a string in the contract state storage, which points to a resource attachment stored in IPFS. The earlier tests were replicated using event logs for registering batches, employing the `publish-Batch` method of the `Step-Contract`, which emits the event `emitBatch` during batch registration. Furthermore, assuming that 32 characters are sufficient for the attributes listed in Table 1, the data type `bytes32` was utilized for the variables in the `Step-Contract` as well as for the MultiHashes related to the attachments. This choice is based on the fact that fixed-length byte variables are more efficiently managed than strings. As reported in Table 3(b), utilizing events/logs considerably lowers storage costs.

Regarding the effectiveness of introducing IPFS alongside traditional blockchains, studies, such as [2], demonstrate that combining IPFS with blockchain can enhance data storage efficiency, in terms of time and space, thus providing a more flexible architecture for decentralized applications.

5 Consumer-Centric Implications of the Resource-Oriented Approach

Section 4 presented quantitative results demonstrating the technical feasibility and cost-efficiency of implementing a resource-oriented blockchain integration approach in wine supply chains. Building on these empirical findings, this section examines the broader implications of this approach, with a focus on consumer-centric aspects. The effectiveness of supply chain innovations is ultimately measured by their ability to deliver value to end consumers. Therefore, this discussion analyzes how the resource-oriented approach may influence consumer interactions with wine product information. The resource-oriented methodology, utilizing a base data model with specific views, offers potential advantages for standardizing information access across diverse blockchain technologies. This standardization could enable consistent consumer interaction with product data, irrespective of the underlying blockchain infrastructure used by different wine producers. However, the implementation of this approach may face several challenges. These include ensuring data integrity across multiple supply chains, addressing producers privacy concerns, and striking a balance between information depth and user interface simplicity.

Standardization and Customization of Information. The base data model with specific views facilitates the creation of customized information presentations tailored to diverse consumer needs. This flexibility enables the development of user interfaces that can cater to various consumer segments, from casual wine consumers to connoisseurs. The ability to present relevant information while obscuring complex supply chain details may enhance user engagement and comprehension.

Traceability and Trust. The resource-oriented approach potentially enhances product traceability, allowing consumers to access detailed information about a wine's journey from production to point of sale. This increased transparency may contribute to building consumer trust and confidence in product authenticity and quality claims.

Privacy and Information Control. The view mechanism allows for granular control over shared information, potentially addressing privacy concerns while still providing valuable data to end-users. This balance between transparency and privacy protection is crucial for consumer acceptance and regulatory compliance.

Extensibility and Future Adaptability. The extensibility of this approach is noteworthy. As consumer needs evolve, new views can be created from the base model without necessitating changes to the underlying blockchain structure. This flexibility facilitates the rapid introduction of new features or information types that consumers may find valuable.

Integration with Third-Party Applications. The resource-oriented approach may simplify integration with third-party consumer applications, such as wine rating platforms or food pairing services. This integration potential could further enrich the consumer experience and expand the utility of the blockchain-based information.

Challenges and Limitations. Despite these potential benefits, there are challenges to be considered. For example, ensuring data accuracy and consistency across multiple supply chains presents a significant technical and operational challenge, as it does determining the appropriate depth of information to provide.

6 Concluding Remarks

In this paper, a methodological approach to design a resource-oriented architecture for BCT integration on top of intertwined supply chains has been proposed. The approach is organised into BCT-independent steps, where resources within the supply chain and *RESTful Services* to share them are modelled, and BCT-dependent steps, where RESTful services are mapped to Smart Contracts of specific BCTs. Resource modelling promotes a homogeneous representation of supply chain data in BCT-based Web Applications for traceability and better coherency across different use cases. Experiments demonstrated that the resource-oriented approach simplifies the business logic of Smart Contracts, positively affecting storage costs.

The resource-oriented architecture design offers an alternative to the service-oriented vision on blockchains and Smart Contracts [5,8,14]. The latter typically involves a broader set of Smart Contracts and thus higher development costs, being suitable for modelling complex workflows requiring complex business logic. However, for scenarios like supply chains in the agri-food domain the resource-oriented perspective brings the aforementioned advantages. Furthermore, our approach aims at enhancing consumer engagement and trust in the wine industry. By providing a standardized method for accessing and interpreting product data across different blockchain technologies, it offers consumers a consistent and transparent view of wine supply chains through Web applications, potentially facilitating more informed decision-making. However, successful implementation will require addressing challenges such as ensuring data accuracy across multiple supply chains and balancing information depth with user interface simplicity.

Our upcoming research will focus on exploring the integration of alternative blockchain technologies, particularly IOTA [19], which has been specifically engineered for Internet of Things (IoT) data management—a critical component in Agriculture 4.0 and modern food industry applications. We also plan to extend our approach to other application domains and conduct empirical studies on consumer interaction with BCT-based Web applications, including strategies for overcoming adoption barriers within the wine industry. Finally, building upon the work of [7], we plan to develop software component design patterns and establish a decentralized component registry.

References

1. Repository with supplementary material. https://anonymous.4open.science/r/WISE2024-BCT
2. Alizadeh, M., Andersson, K., Schelén, O.: Efficient decentralized data storage based on public blockchain and IPFs. In: 2020 IEEE Asia-Pacific Conference on Computer Science and Data Engineering (CSDE), pp. 1–8 (2020)
3. Bagozi, A., Bianchini, D., De Antonellis, V., Garda, M., Melchiori, M.: A blockchain-based approach for trust management in collaborative business processes. In: Zhang, W., Zou, L., Maamar, Z., Chen, L. (eds.) WISE 2021. LNCS, vol. 13080, pp. 59–67. Springer, Cham (2021). https://doi.org/10.1007/978-3-030-90888-1_5
4. Bianchini, D., De Antonellis, V., Garda, M., Melchiori, M.: Resource-oriented approach for effective blockchain integration in intertwined supply chains. In: International Conference on Database and Expert Systems Applications (DEXA 2024), pp. 18–33 (2024)
5. Daniel, F., Guida, L.: A service-oriented perspective on blockchain smart contracts. IEEE Internet Comput. **23**(1), 46–53 (2019)
6. Eberhardt, J., Tai, S.: On or off the blockchain? Insights on off-chaining computation and data. In: Service-Oriented and Cloud Computing: 6th IFIP WG 2.14 European Conference (ESOCC 2017), pp. 3–15 (2017)
7. Falazi, G., Breitenbücher, U., Daniel, F., Lamparelli, A., Leymann, F., Yussupov, V.: Smart contract invocation protocol (SCIP): a protocol for the uniform integration of heterogeneous blockchain smart contracts. In: Dustdar, S., Yu, E., Salinesi, C., Rieu, D., Pant, V. (eds.) CAiSE 2020. LNCS, vol. 12127, pp. 134–149. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-49435-3_9
8. Falazi, G., et al.: Unified Integration of Smart Contracts through Service Orientation. IEEE Softw. **37**(5), 60–66 (2020)
9. Ferrández-Pastor, F.J., et al.: Agricultural traceability model based on IoT and blockchain: application in industrial hemp production. J. Ind. Inf. Integr. **29**, 100381 (2022)
10. Hader, M., et al.: Applying integrated blockchain and big data technologies to improve supply chain traceability and information sharing in the textile sector. J. Ind. Inf. Integr. **28**, 100345 (2022)
11. Jabbar, S., et al.: Blockchain-enabled supply chain: analysis, challenges, and future directions. Multimedia Syst. **27**, 787–806 (2021)
12. Khan, S., et al.: Blockchain technologies as enablers of supply chain mapping for sustainable supply chains. Bus. Strateg. Environ. **31**(8), 3742–3756 (2022)
13. Kramer, M.P., Bitsch, L., Hanf, J.: Blockchain and its impacts on agri-food supply chain network management. Sustainability **13**(4), 2168 (2021)
14. López-Pintado, O., et al.: Caterpillar: a business process execution engine on the ethereum blockchain. Softw. Pract. Exp. **49**(7), 1162–1193 (2019)
15. Motta, G.A., Tekinerdogan, B., Athanasiadis, I.N.: Blockchain applications in the agri-food domain: the first wave. Front. Blockchain **3**, 6 (2020)
16. Patelli, N., Mandrioli, M.: Blockchain technology and traceability in the agrifood industry. J. Food Sci. **85**(11), 3670–3678 (2020)
17. Qian, J., et al.: Food traceability system from governmental, corporate, and consumer perspectives in the European Union and China: a comparative review. Trends Food Sci. Technol. **99**, 402–412 (2020)

18. Ravi, D., et al.: Privacy preserving transparent supply chain management through hyperledger fabric. Blockchain: Res. Appl. **3**(2), 100072 (2022)
19. Reyna, A., Martín, C., Chen, J., Soler, E., Díaz, M.: On Blockchain and its Integration with IoT. Challenges and opportunities. Future Gener. Comput. Syst. **88**, 173–190 (2018)
20. Román-Martínez, I. et al.: Blockchain-based service-oriented architecture for consent management, access control, and auditing. IEEE Access **11**, 12727–12741 (2023)
21. Vern, P., Panghal, A., Mor, R.S., Kamble, S.S.: Blockchain technology in the agrifood supply chain: a systematic literature review of opportunities and challenges. Manag. Rev. Q. 1–33 (2024)