smart cities

MDPI

Article

Blockchain Orchestration and Transformation for Construction

Mohammad Darabseh * and João Poças Martins

CONSTRUCT–GEQUALTEC, Faculty of Engineering (FEUP), University of Porto, Rua Dr. Roberto Frias s/n, 4200-465 Porto, Portugal

* Correspondence: darabseh@outlook.com

Abstract: Blockchain-related studies that focus on solving AECO (Architecture, Engineering, Construction and Operation) digital management environment issues, such as data protection and data ownership, show the projected benefits of Blockchain-based digital construction environments. However, adopting such technology will require a holistic approach to ensure it does not result in data redundancy, leading to digital system inefficiencies. This article studies the Blockchain construction synergies from the infrastructure point of view to understand its future in construction. The article visualises Blockchain infrastructure elements and fits them within the construction project's digital environment. A novel framework for Blockchain orchestration and implementation and a blueprint for developing Blockchain applications for construction are presented. The proposed blueprint is then used to develop a Blockchain application using Hyperledger Firefly. The article builds on the previous literature and Blockchain applications on the Ethereum public Blockchain. The expected benefit of such a framework is providing a practical perspective on the implementation side of Blockchain in construction.

Keywords: Blockchain; AECO; construction; NFT

Citation: Darabseh, M.; Poças Martins, J. Blockchain Orchestration and Transformation for Construction. *Smart Cities* **2023**, *6*, 652–675. https://doi.org/10.3390/smartcities6010031

Academic Editors: Miguel Pincheira and Massimo Vecchio

Received: 31 January 2023
Revised: 10 February 2023
Accepted: 17 February 2023
Published: 20 February 2023

1. Introduction

Previous studies have suggested that Blockchain technology can contribute positively to the digital construction environment by introducing trustless interaction between the involved parties regardless of their intentions or trustworthiness [1].

Blockchain implementation in the construction industry can be viewed from two perspectives: long- and short-term. From a long-term perspective, Blockchain can be used by construction-related entities to provide overall control or oversee digital data for the assets that fall under the entity's reach. In the short term, Blockchain can provide authentic and reliable information for the project stakeholders, thus contributing towards fulfilling the project aims. Short-term Blockchain implementations provide more extensive data to track a specific data set. In contrast, long-term implementations provide an abstracted version of data collected from multiple sources of interest for the entity.

Both implementations should be interoperable to avoid data redundancy and maximise the benefits of adopting a Blockchain-based system. In order to achieve the ideal configuration, this article presents a Blockchain implementation and orchestration framework for construction operations and project management.

The article consists of two main parts: Section 2 examines the required infrastructure for developing Architecture, Engineering, Construction, and Operation (AECO) Blockchain applications. Section 3 discusses AECO assets in digital environments and managing AECO assets with Blockchain.

Blockchain adoption for AECO processes can help expand operations automation, reducing the cost, time and resources needed. In addition, Blockchain ledgers can provide auditable event logs for projects for traceability and accountability purposes [2].

This study is designed to serve as a general guideline for developing AECO Blockchain Applications from a technical point of view by analysing Blockchain technology components ontologically. The presented work shows the components and links them together

in a reference model. In addition, a blueprint for developing AECO based on the current de facto Blockchain software development approach customised to fit the AECO industry is presented. Later, a demo for managing AECO assets using Hyperledger Firefly is presented. The article uses real-world public Blockchain applications, mainly on the Ethereum Blockchain used by millions worldwide to manage and transfer financial assets, to synthesise the proposed blueprint. Applications include (1) Uniswap [3]; (2) OpenSea [4]; (3) Chainlink [5]; (4) The Graph [6]; (5) Aave [7]; (6) Curve [8]; (7) Decentraland [9]; (8) Axie Infinity [10]; (9) The Sandbox [11]; (10) Ethereum Name Service (ENS) [12].

2. Blockchain and Construction

The discussion of Blockchain applications in the AECO industry takes different approaches and directions, depending on the purpose of the proposed application. However, it is essential to choose the proper infrastructure that satisfies the needs of projects and companies to overcome the technical, cultural, and legal barriers to adopting Blockchain.

Blockchain in the construction-related literature is approached from two viewpoints: as a construction General-Purpose Technology (GPT) [13] or as a construction-enabling technology [14]. Blockchain in standalone implementation or synergy with other construction technologies, such as Building Information Modeling (BIM), shows numerous benefits for the industry. In the literature, Blockchain was investigated with the construction supply chain [15–17], construction project information to improve information flow [17,18], information exchange [19], information authenticity [20] and information redundancy [21]. Blockchain is also seen as one of the Construction 4.0 pillars and it has been investigated alongside other enabling technology to reach the industry goals toward sustainability [22–24]. The security and immutability features in Blockchain-based solutions were utilised in several studies to help the industry combat the growing cybersecurity risks [25–27].

2.1. Software Architecture

Several studies have attempted to find the most suitable Blockchain implementation for construction projects among the three types: (1) public, (2) private, and (3) consortium [28]. However, the AECO industry is multidisciplinary, and it is hard to fit all under one category as each application aims for a specific purpose. Therefore, a monolithic software architecture that encapsulates all the subsystems [29] does not fit the AECO industry when adopting Blockchain [29].

In comparison, the components of a microservices architecture system are designed to function independently while maintaining connectivity with each other through the Application Programming Interfaces (APIs) [30]. The microservices architecture pattern offers flexibility, scaling and independence without the obligation to develop one solution that fits all. Adopting the microservices allows developers to customise each application to achieve its goals without compromising security, costs, privacy or efficiency. Figure 1 illustrates the difference between monolithic architecture and microservices architecture.

Storing all the data on public Blockchains is not practical because of accruing transaction costs; however, data integrity can be assured using data privacy and integrity preservation techniques such as Zero Knowledge Proofs (ZKPs) and data sampling.

2.2. AECO Blockchain Infrastructure (AECO-BI)

Blockchain systems are designed as a sandbox environment, meaning the smart contracts run in isolation from the network resources in a virtual machine. An example is the Ethereum Virtual Machine (EVM) [31], the virtual machine for the Ethereum Blockchain. The use of such a structure ensures network stability and security.

In addition, Blockchain systems are deterministic. Here, determinism means that the block produced by a network node should be identical to the block produced by other nodes, with the transactions ordered identically inside the block. Blockchains achieve

determinism by using a consensus algorithm such as the Proof-of-Stake (PoS) [32] used in the Ethereum Blockchain.

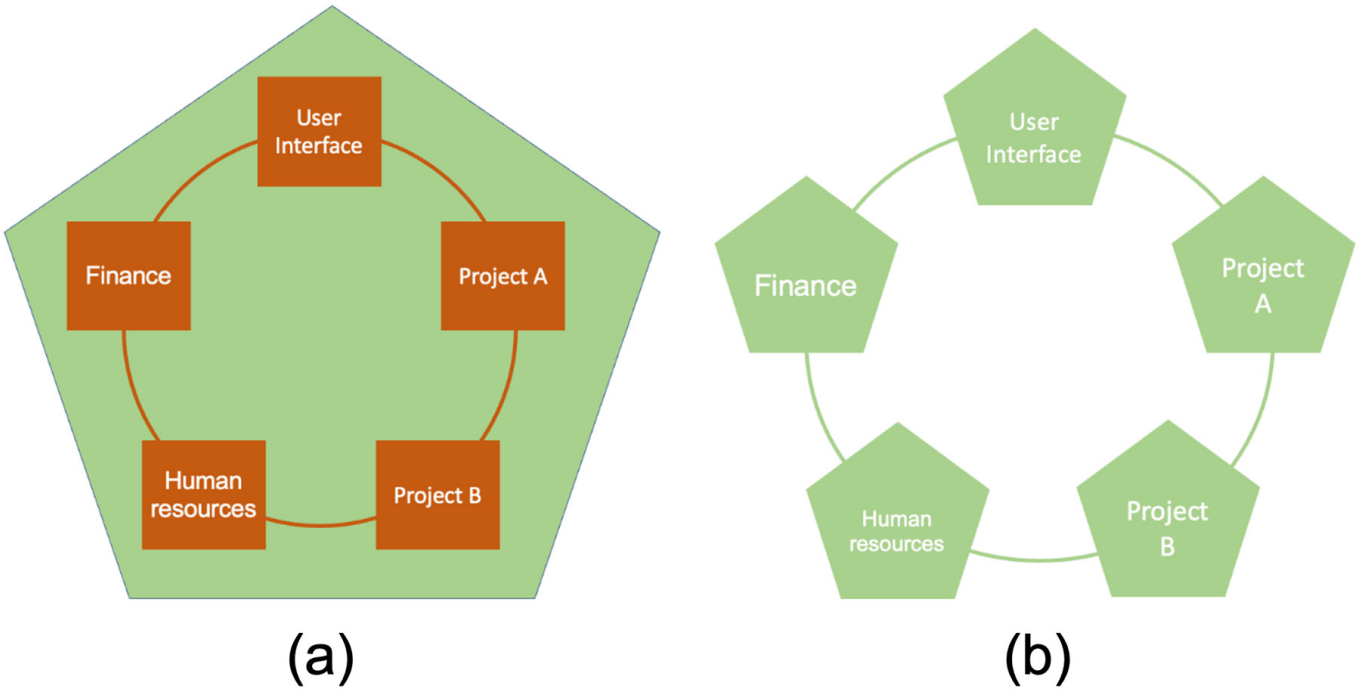

Figure 1. (**a**) Monolithic architecture. (**b**) Microservice architecture.

While the expected benefits of adopting a Blockchain-based system to manage construction-related activities seem promising, such as improved collaboration and data consistency, Blockchain isolation and determinism can throttle the efficiency of such systems. Blockchains perceive data in two groups: (1) on-chain data stored in the Blockchain ledger; (2) off-chain data, which refers to any data or knowledge available outside the isolated Blockchain network environment. The AECO industry activities occur in real life, where Blockchain networks are isolated for security and determinism purposes. Therefore, Blockchain systems for AECO-related purposes should be able to fetch off-chain data without compromising the integrity of the Blockchain network.

2.2.1. Blockchain Oracles

Blockchain smart contracts are designed as an isolated environment to guarantee their functionality and objectivity. However, this configuration hinders Blockchain adoption for two reasons: (1) the limited data available on-chain; (2) the expensive on-chain computation. Oracles emerged to overcome these obstacles. Blockchain Oracles is the term used to describe the bridges that allow Blockchains to access off-chain data sources. Oracles are information channels designed to provide smart contracts with real-world information needed to help them achieve their objectives [33]. Oracles have three main responsibilities: (1) collect data from off-chain sources; (2) transfer data; (3) make data available on-chain by storing it in a smart contract. Modern construction industry data is stored in digital format; however, it is not available for smart contracts directly. Figure 2 shows an illustration of the working principle of a Blockchain oracle.

It should be noted that an oracle is not a data source but a Blockchain compatible connection with an off-chain data source. As such, this solution does not compromise the Blockchain network stability or its determinism by relaying the requested information to another smart contract to achieve implied determinism for the main smart contract.

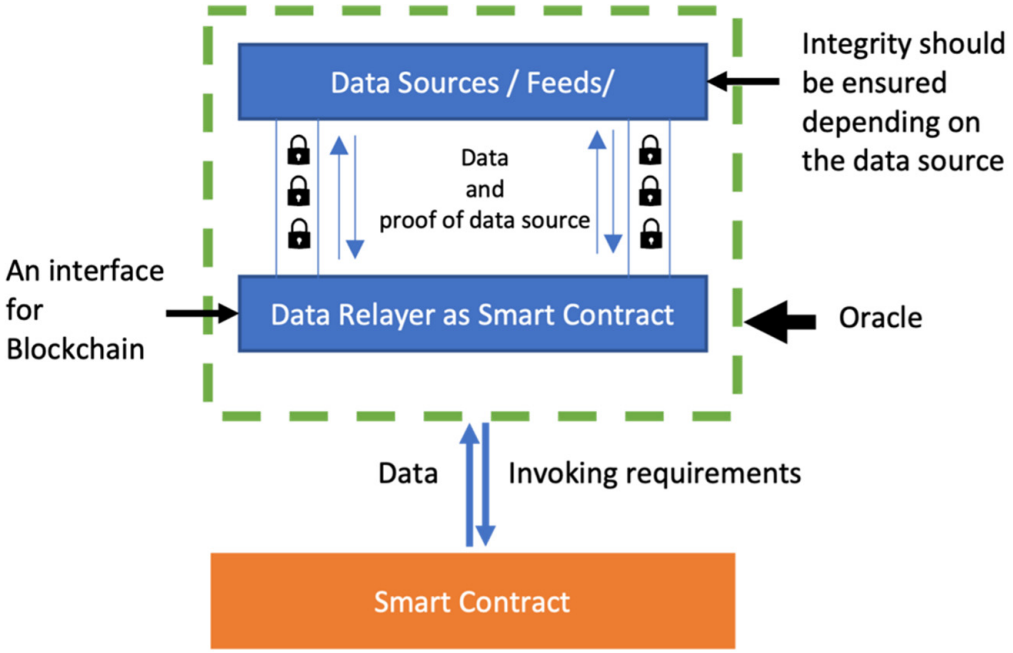

Figure 2. Blockchain Oracles Working Principle.

Oracles can be classified according to their source of information into three categories: (1) software oracles where the information is relayed to the smart contracts from software such as BIM authoring tools or project management software; (2) hardware oracles where the source of information is an Internet of Things (IoT) device, such as a biometrics sensor, which acts as a timekeeper in a construction project; (3) human oracles, where the data is posted by a human actor, such as a construction project consultant.

Oracles can also be divided into three categories according to the information flow direction: (1) inbound, where the purpose of the oracle is to receive information from an external source [34]; (2) outbound, where the purpose of the oracle is to send information to an external source; (3) inbound/outbound oracles, where the oracle can communicate in both directions. Figure 3 below illustrates this classification.

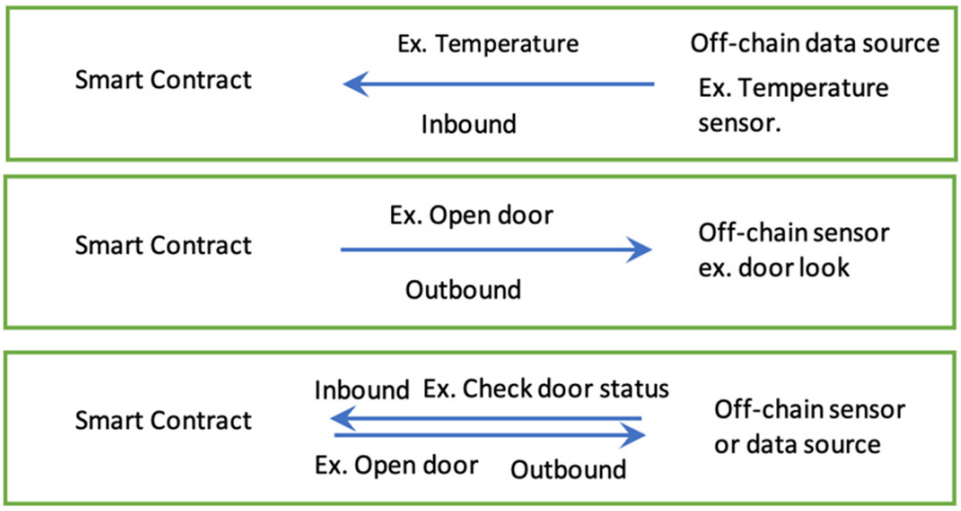

Figure 3. Oracles Types Based on Information Flow Direction.

An oracle can be classified as centralised or decentralised, depending on the data source. If the oracle provides data using a single feed, it is considered a centralised oracle. Conversely, an oracle that provides data by fetching multiple sources to generate the requested value is considered decentralised.

Depending on their deployer, oracles can be self-hosted or outsourced.

Oracles can be classified into two types according to their main objective: carrier oracles fetch data from off-chain sources; while computation oracles execute a computational process for a smart contract off-chain when it is not possible or feasible to do it on-chain due to computation costs.

Figure 4 summarises the oracles classifications presented in this section [33].

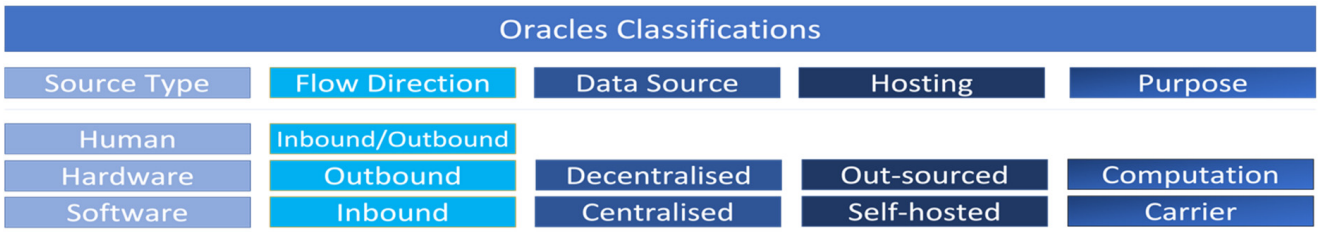

Figure 4. Oracles Classification Summary.

2.2.2. AECO Blockchain Ecosystem

An adequate infrastructure is needed to achieve efficient Blockchain implementation in construction projects and operations. While the elements required for developing a Blockchain solution vary depending on its purpose, there are two fundamental components: (1) Blockchain network and ledger; (2) Blockchain oracles. The network is responsible for the data integrity stored in the ledger by enforcing consensus. Oracles help AECO projects and activities communicate with real-life information feeds to help smart contracts execute their functions based on valid data.

A reference model, illustrated in Figure 5, was developed to help visualise the AECO Blockchain ecosystem. The model separates the Blockchain system into two layers. The base layer (called the technology layer) includes the physical infrastructure components, such as the machines acting as Blockchain nodes. This layer includes two logical component blocks: The security block refers to the Blockchain core services and capabilities that give Blockchain unique features, such as the immutable ledger, smart contracts, and Blockchain-based identities that allow identifying users and their transactions. The perception block contains supplementary technologies that aid the smart contracts with external information from off-chain sources such as IoT sensors or a legacy information system, or provide solutions for computational tasks using technologies such as Artificial Intelligence (AI) or Machine Learning (ML). In addition, the perception block includes Blockchain bridges which refer to special oracles that establish inter-Blockchain connectivity. The various applications for Blockchain could lead to several parallel technology layers to satisfy each purpose without compromising the system's security or efficiency. However, multiple technology layers could lead to data fragmentation or data segregation.

Hence, another global layer is required to oversee the Blockchain systems for construction projects and operations. The manifestation layer presents an omniscient view of multiple underlying Blockchain systems. The main component of the manifestation layer is the supernode. A supernode is a node connected to multiple Blockchains simultaneously, allowing it to interact with its smart contracts and propose transactions. The AECO supernode oversees all the construction and operations of on-chain activities, smart contracts, participants, and other off-chain systems such as storage and oracles. The supernode allows users to manage their Blockchain presence through namespacing. An engineer, for example, can exist in the construction site and the design office namespaces, even if they do not exist on the same Blockchain.

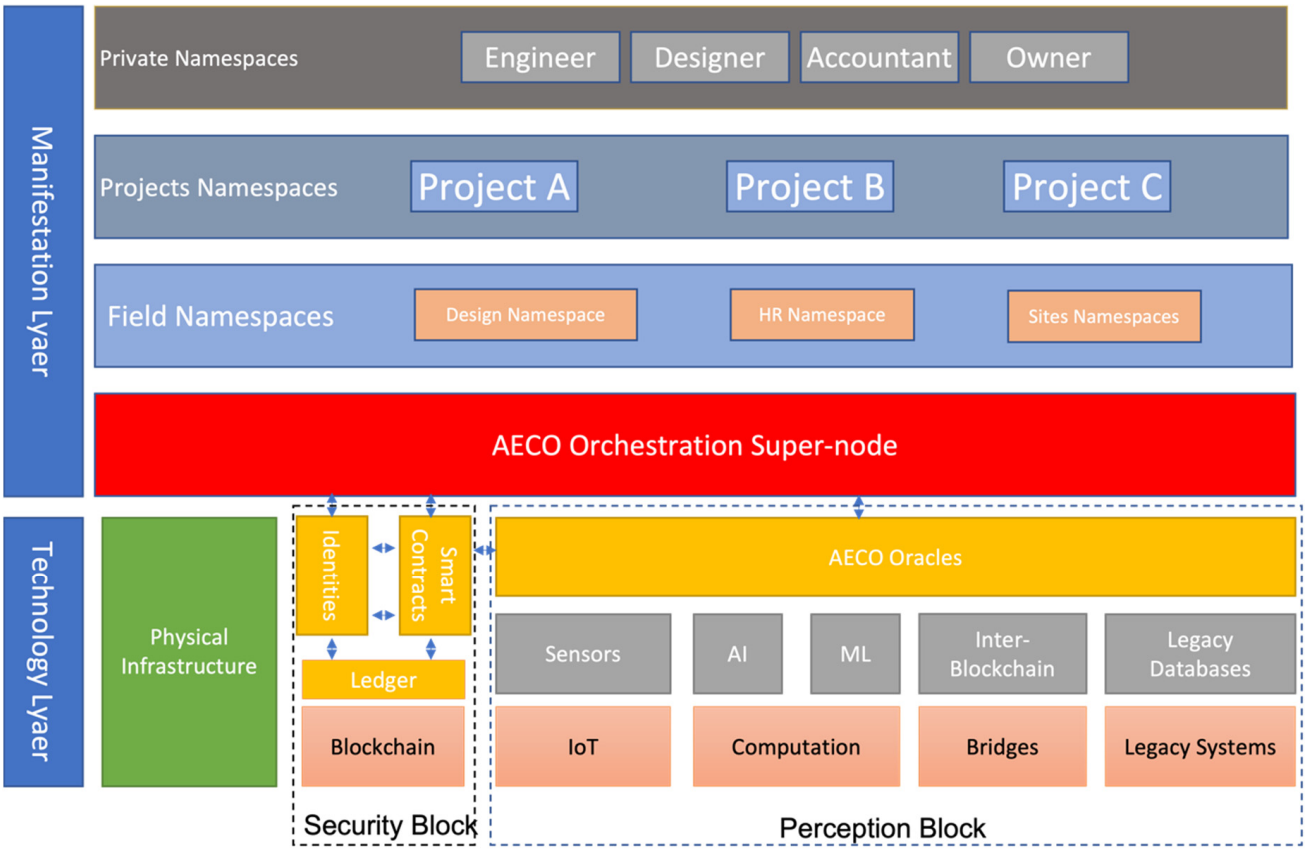

Figure 5. Reference Model for AECO System Elements.

The AECO orchestration supernode utilises the microservice structure to create multiple data buses between the system components to achieve several on-chain or off-chain connectivities. The supernode can provide additional features, such as APIs, private messaging, and distributed storage such as InterPlanetary File System (IPFS), for an improved Blockchain experience. Each namespace [35] represents a Blockchain environment. The supernode service is not a Blockchain but a connection orchestration service to multiple ones. Establishing a Blockchain network, Blockchain oracle, or distributed storage system is a separate process and infrastructure. Therefore, the supernode-based architecture allows for the use of multiple infrastructures based on the needs of the solution without requiring a one-size-fits-all solution.

2.2.3. AECO Smart Contracts

Smart contracts are the core service of modern Blockchain systems. They have enabled the advanced utilisation of a Blockchain network through definitions and functions to track data and enforce rules.

Smart contracts are deterministic [36]. Deterministic algorithms always return the same output if the same input is used an indefinite number of times. The concept applied to a Blockchain reflects the Blockchain's main value proposition, which is a single source of truth. There are two levels of determinism in Blockchain: Network determinism and external data determinism. Network determinism is how the Blockchain processes data in the ledger to output the same value every time to reach a consensus. The common approach to reaching network determinism is the order-execute architecture [37]. The order-execute architecture removes non-deterministic operations from smart contracts using a domain-specific programming language for writing smart contracts; an example of order-execute architecture Blockchains is Ethereum. The difficulty in achieving determinism when writing smart contracts using traditional programming languages motivated the

Ethereum Blockchain to achieve determinism by creating the Ethereum Virtual Machine (EVM) and multiple specific high-level programming languages such as Solidity. A smart contract is first written in Solidity and then compiled to bytecode to make it understandable by the EVM. After that, the bytecode of the smart contract is deployed to the Blockchain and stored in the ledger. Figure 6 is an illustration of Ethereum order-execute architecture.

Figure 6. Ethereum Order-Execute Architecture.

Another way to reach determinism is the execute-order-validate architecture. In such architecture, the transaction is executed before sending it to the ordering node to check its correctness and ensure its determinism. An example of Blockchains that use such architecture is Hyperledger Fabric. Smart contracts in Fabric are written using general programming languages such as Java, Go, and JavaScript [38].

The relevance of oracles for construction was discussed in the previous sections, as they provide a secure channel to bridge off-chain AECO data to Blockchain networks. Additionally, oracles ensure external data determinism. When developing an AECO smart contract, it is important to define the off-chain data required to establish the needed Oracle infrastructure. The flowchart in Figure 7 simplifies the data onboarding process for smart contracts while maintaining its determinism.

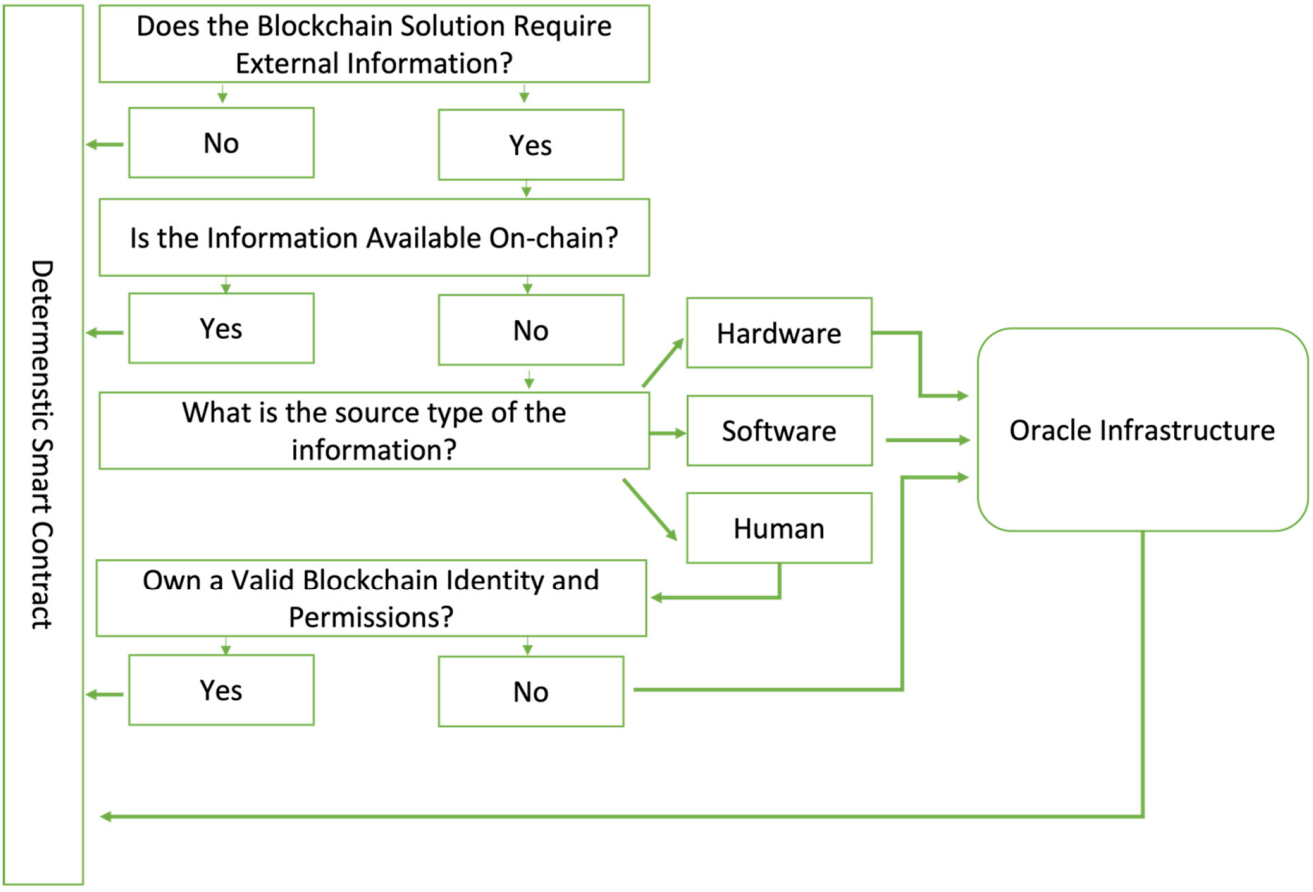

Figure 7. Deterministic Data Onboarding Process.

2.2.4. Blueprint for Developing AECO Blockchain Solutions

The spectrum for Blockchain AECO applications is wide and multidisciplinary. Blockchain is an enabling technology that can be used with other domain-specific technologies, such as BIM, or merged with other management approaches to help it reach its goals, such as Integrated Project Delivery (IPD) [39]. However, every solution that uses Blockchain shares common components that help Blockchain function properly. This section provides a guideline that can be used to develop Blockchain AECO applications. The proposed blueprint shown in Figure 8 provides six steps to develop a Blockchain-based application.

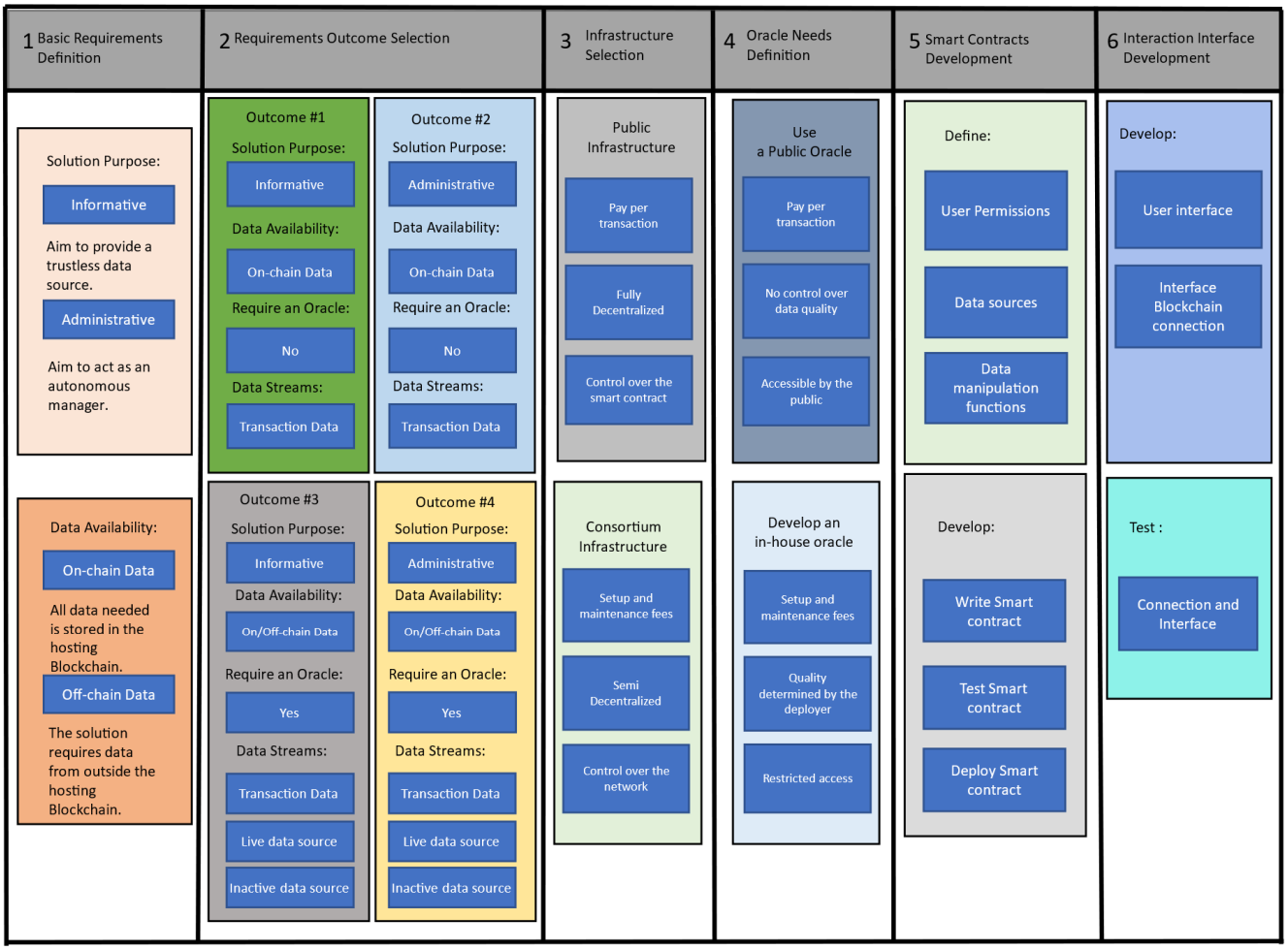

Figure 8. Blueprint to Develop AECO Blockchain Applications. Four outcomes based on the basic reequipments definition.

1. Basic requirements

Step one is the basic requirements definition, where two aspects are clarified. First is the solution purpose, which can be (1) informative, when the sole purpose of the Blockchain solution is to act as a trustless data source, or (2) administrative, when the solution will take some management roles and act as an autonomous manager. Therefore, an administrative solution has a wider scope than an informative one. The Second aspect is the solution data availability, which refers to the location of the data, as presented in Section 2.2. On-chain data is hosted directly on the Blockchain, while off-chain data is used when the application requires information from sources other than the Blockchain ledger. This first step results in four possible outcomes as a combination of the solution purposes and data availability.

2. Requirements outcomes

The second step is to pick the most suitable outcome for the proposed application.

Outcome one is an informative and on-chain-data-only application. These applications do not require developing an oracle, as they only use the transaction data stored in the ledger.

Outcome two is an administrative application; however, it is restricted to on-chain data. The main difference between outcomes one and two is the functions in the smart contracts. While outcome one smart contracts can query information, outcome two has functions to query and update the information stored in the smart contract.

Outcomes three and four use both on- and off-chain data. As such, oracles are required to communicate with off-chain data sources, including live data sources such as APIs, and inactive data sources such as files stored on an off-chain storage source. However, outcome three is informative, which means the smart contract does not include functions for actions beyond collecting information from on- and off-chain sources.

Outcome four represents the most advanced capabilities that can be delivered by a Blockchain application where all data sources are included. The smart contract can collect information and execute orders based on predefined conditions.

3. Infrastructure

The third step is infrastructure selection based on the previous steps and the application needs. Depending on the application, two types of infrastructure can be used: (1) Public Blockchain infrastructure, which refers to Blockchains that use public nodes to store their ledger and execute transactions. Public Blockchains are fully decentralised, and payment occurs per transaction; however, the deployer has control over the smart contract only, not the network itself. (2) Consortium Blockchain infrastructure, which refers to Blockchains hosted and operated by a group of participants who control the whole network. These networks require setup and maintenance fees and are considered semi-decentralised as the data is stored in a limited number of nodes. However, the control over the whole network provides privacy for the data stored in the ledger.

4. Oracle needs

The fourth step is for outcomes three and four, where off-chain data is used. The oracle-needs-definition step aims at deciding the suitable options for establishing data and communication bridges with AECO assets related to the Blockchain application. There are two options: (1) public oracles and (2) in-house oracles. Public oracles are not deployed by the application deployer and are accessible to the public. In-house oracles are deployed and maintained by the application deployer [40].

The application cannot control public oracle data quality; however, the user-only pay-per-transaction and maintenance is the responsibility of the oracle deployer. Public oracles currently cover a limited number of feeds and a specific-purpose AECO Blockchain application will require deploying an in-house oracle to satisfy its off-chain data needs.

5. Smart contract deployment

Step five is smart contract deployment, consisting of two processes: defining and developing the smart contract. The smart contract holds the logic, the rules and the authority to regulate the Blockchain application processes. An application can consist of one or more smart contracts, depending on its purpose. The smart contract definition process defines the user and user permissions, data sources and data manipulation functions. The smart contract development process includes writing and testing the smart contract and deploying it to the infrastructure chosen in step three.

6. Interaction interface development

The last step is the interaction interface development. This step focuses on developing an interface to lower the technical capabilities required from the user to interact with a

Blockchain-based application. This step has two processes: developing the user interface and the interface Blockchain connection, then testing the connection and interface.

3. AECO Assets Management Supernode

The previous sections discussed the Blockchain development options for AECO-related applications. This section focuses on the options available to develop Blockchain applications. Any element contributing to completing an AECO objective is considered an asset. The proper management of assets can provide an overall improvement for the objective itself. The AECO industry is assets-oriented; therefore, improving AECO assets management can enhance objectives' delivery and execution.

3.1. AECO Assets

In digital environments, AECO assets can be categorised into two groups: (1) AECO digital assets, which refer to those constructed and stored digitally. (2) AECO digitalised assets refer to real-life physical assets represented digitally by a model and controlled through a digital communication protocol [41]. The digital twin concept aims to create a digital replica of a physical asset to manage and improve its utilisation throughout its lifecycle.

Blockchain usage can be extended to represent AECO assets in fungible or non-fungible form. Blockchain improves AECO asset utilisation by creating an authentic asset with usage trail and ownership to provide a comprehensive overview of the asset in focus. There are two types of Blockchain-based assets: (1) fungible tokens and (2) non-fungible tokens.

3.1.1. Fungible Tokens (FTs)

Certain AECO assets can be presented on the Blockchain as fungible tokens. Fungible tokens are used to present a common construction asset. A common construction asset does not have extra attributes distinguishing the asset from another asset from the same class. An example of such assets in AECO is construction project consumables such as standard Portland cement bags. While the product is unique, each bag has no difference, making it fungible. Using fungible tokens to track AECO assets can benefit construction projects during the delivery and operation phases by providing accurate information about inventory for better control over the resources assigned for these projects. In addition, it provides an indicator of the quality of the used products. For example, a lamp replacement pattern can be visualised based on the lamp token burn rate, where the token burn rate can be used as of indication of a damaged asset [42].

3.1.2. Non-Fungible Tokens (NFTs)

Practically all AECO assets can be presented on the Blockchain as NFTs. However, the approach varies depending on the purpose of presenting the asset. For example, a building digital twin model can be presented in a holistic approach in one NFT or as a set of NFTs in a modular digital twin approach [43].

NFTs are differentiated from FTs mainly by being unique instead of common. NFTs' other characteristics are (1) standardised, where an NFT represents a unique object using a common format. Standardisation helps develop solutions to read and utilise these NFTs within information systems; (2) ownable, where an NFT always belongs to an owner, who can be permanent or temporary. (3) transferable or account bounded token: An NFT ownership model should define whether transferring the token to a new owner is possible. Non-transferable NFTs are called account bonded tokens (ABTs). ABTs can be used in AECO applications to restrict an NFT's usage spectrum. ABT examples include the following: Know Your Customer (KYC), where the KYC NFT is a special purpose on-chain token identity; university degrees are another example of ABTs where the university issues an ABT NFT to certify an academic achievement to the wallet owner. When the ABT is issued, it is transferred to the owner's wallet and cannot be moved to another

wallet. (4) Static or dynamic refers to the data identifying the NFT's unique attributes. The NFT can be dynamic or static, depending on the NFT's purpose. For example, an NFT representing a window specification is static, and these specifications will not change over time. Conversely, a building's automated entrance opening schedule NFTs will be dynamic as the schedule might change on special occasions or for emergencies. (5) NFTs contain metadata which are partially stored on-chain and partially on an off-chain medium. (6) Refers to a data source where every NFT uses its on-chain metadata to point at an off-chain data source for a detailed NFT record. The ERC-721 is the Ethereum standard that provides guidelines for the NFTs' creation process. Figure 9 below illustrates the NFT components according to the ERC-721 standard [44].

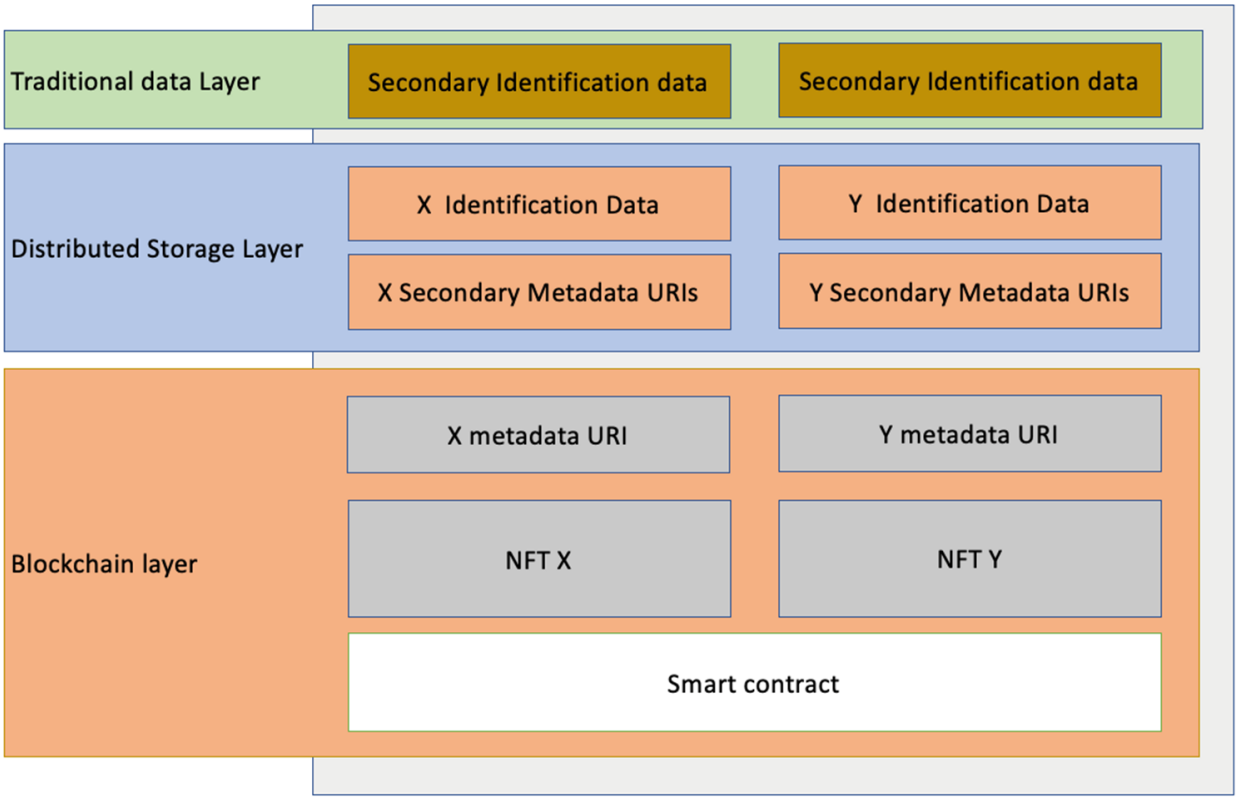

Figure 9. NFT Components According to ERC-721.

According to the ERC-721 [44], a smart contract is used to create NFTs, acting as a factory. Depending on the functions of the smart contract, it can create one type of NFT or more. Further, it can make NFTs of the same type, but each has at least one unique attribute that differentiates it from the rest. The generated NFT using the smart contract on-chain metadata can vary; however, all NFTs that follow the ERC-721 have a Universal Resource Identifier (URI) that points to the metadata file stored off-chain in the distributed storage protocol IPFS. The metadata stored in the IPFS node are serialised data in JSON format. The JSON file contains the NFT identification information and, in certain assets, extra URIs related to the NFT for data stored on IPFS or traditional storage infrastructure [45].

3.1.3. NFTs for AECO Applications

An NFT attests to the Blockchain system's capability to produce unique digital assets that can be distinguished from fake replicas. NFTs act as proof of authenticity or a tracker for an asset lifecycle. This section explores the NFTs application in the AECO industry.

An NFT can serve one or multiple purposes based on its creator's intentions and the environment where it is supposed to function. This means that an NFT's purpose can

change based on the context. For example, a project manager might use an NFT to track a project objective while the engineer uses the same NFT as proof of experience in his resume.

There are four groups of NFT AECO applications: (1) NFT as a representation of an AECO objective; (2) NFT as an AECO document; (3) NFT as an AECO certificate; (4) NFT as AECO asset representation.

7. NFT as a representation of an AECO objective

NFT as a representation of an AECO objective is the use of NFTs to represent an agreed AECO goal. An AECO objective NFT can act as a reference to the objective team regarding the objective specification and status. An AECO objective NFT is a permanent record of the objective, which can protect participants from unfair requests or delay in payments; such an NFT can be used as a progress report.

8. NFT as an AECO document

Regardless if the AECO document is in a digital or digitised document format, the NFT connected to it is a Blockchain extension of the original document. It provides a detailed document record, including data such as the identity of the issuers, date, and purpose. This data can help verify the document authenticity or track the process of issuing such documents, which can be utilised in other aspects, such as improved productivity estimation for processes.

9. NFT as an AECO certificate

NFT as an AECO certificate is similar to NFT as an AECO document; however, the NFT itself is a replacement for the document. NFTs can be used to provide a certificate for an AECO event, structured for events such as confirmation of completion or delay for an AECO process, or a non-structured event such as an internal correspondence. Further, an AECO object can be presented as an NFT. An NFT for an object can provide information about its usage, ownership, and compliance with standards from a quality, safety, or sustainability perspective. NFTs can be used to certify a person, a company, or an entity. Such NFTs can indicate qualifications, assigned tasks, achievements, behaviour, location, or a unique trait that must be presented digitally. An engineer's professional licence NFT issued by the engineering association might be presented in their Blockchain personal wallet to allow the hiring companies to validate their qualifications.

10. NFT as AECO asset representation

NFT as AECO asset representation is a broad aspect where any asset type, regardless of its origin, can be presented on a Blockchain network as an NFT; however, the NFT level of authority over the asset varies depending on the purpose of the NFT. AECO Assets can be categorised based on their existence into digital and physical assets.

They can also be categorised according to their value into financial and non-financial assets. Financial assets are presented as an NFT with authority to control the asset, allowing asset owners to be transferred or escrowed. Such a flexible approach can help AECO entities prove ownership and control the asset smoothly to raise capital or prove liquidity. One asset can be presented as multiple NFTs from different perspectives to satisfy the adequate level of information to consider the NFT valid for that specific purpose. Examples of physical assets are a piece of equipment, a building, or a vehicle. Examples of digital assets are a BIM model, software, or database. Fungible assets could be linked to an NFT if unique attributes emerge for the underlying asset. Table 1 below summarises this section.

3.2. AECO Assets Management Using Firefly

Blockchain-based asset usage in AECO-related environments can bring transparency and knowledge for AECO assets regardless of their importance, which can optimise asset usage throughout projects. In order to properly utilise Blockchain as an AECO asset supervision solution, the blueprint and framework presented in the previous section will be used to develop a Blockchain application to create and exchange AECO design assets as NFTs.

Table 1. Possible Uses of NFTs in the AECO Industry.

NFT Application	Scope	Example
AECO Objective	Project	Project milestone NFT
	Task	Task team assembly NFT
AECO document	Digital document	BIM model NFT
	Digitised document	Mail letter NFT
AECO Certificate	Event	Event completion NFT
	Object	Object ownership NFT
	Process	Process compliance NFT
	Person	Person Qualification NFT
	Entity	Company Classification NFT
AECO Asset	Physical asset	Bulldozer maintenance log NFT
	Digital asset	Software licence NFT
	Financial asset	Land escrow NFT

3.2.1. Defining Application Requirements

The previous section showed the extensive list of assets that can be presented on the Blockchain. For this article's consideration, the application aims to provide information about static digital AECO assets, such as task completion NFT.

Based on the application's aim, the basic requirements are (1) the solution purpose is informative, and it does not require an off-chain data feed. This fits outcome one from the blueprint. (2) The Blockchain infrastructure for this application can be public or consortium, depending on the event's nature and importance. Where events are marked important and involve multiple competing parties, they should use public infrastructure for a fully decentralised experience. The application does not require oracle infrastructure because the assets are static and digital, so there is no need to fetch data from off-chain sources. The smart contract will use the ERC-721 standard for NFTs, and the interaction interface will use Firefly as a Blockchain user interface.

3.2.2. Hyperledger Firefly

Hyperledger Firefly is one of the projects hosted by the Hyperledger Foundation [46]. However, it is not a Blockchain network. Firefly aims to facilitate the creation of Blockchain-based applications by providing abstraction tools that accelerate the application-building process. Firefly is an open-source supernode infrastructure for Web3 services. Web3 refers to the new generation of internet services, whereas Web 1.0 refers to the early web services, where content creation was limited and content was static. Web 2.0 provided more space for the users to create content, and sites provided dynamic content such as social media. In addition, sites provided APIs allowing software to communicate smoothly with the internet [47]. Web3 integrates the latest technologies with the web, such as the semantic web, where content is utilised according to its meaning instead of indexing based on keywords. Other technologies include Artificial Intelligence (AI), Three Dimensions-based websites and services and Blockchain Technology.

Firefly encapsulates the common Blockchain services and components and provides them in a microservice structure to Blockchain application developers. Firefly is a middleware that accelerates the process of building Blockchain applications. Firefly can be used in two patterns: (1) Web3 gateway mode; (2) multiparty mode [48]. Each pattern provides a set of services. The Web3 gateway pattern is directed toward users directly, as it can be used without considering other users' approach to interact with the Blockchain. In contrast, the multiparty pattern requires coordination between the involved parties to create a common Blockchain environment.

Furthermore, the multiparty mode is an advanced Web3 gateway mode with extra features and services. Further, Firefly supports namespaces; therefore, one user can run Firefly in both modes in a separate namespace. Figure 10 shows the services provided by both Firefly patterns.

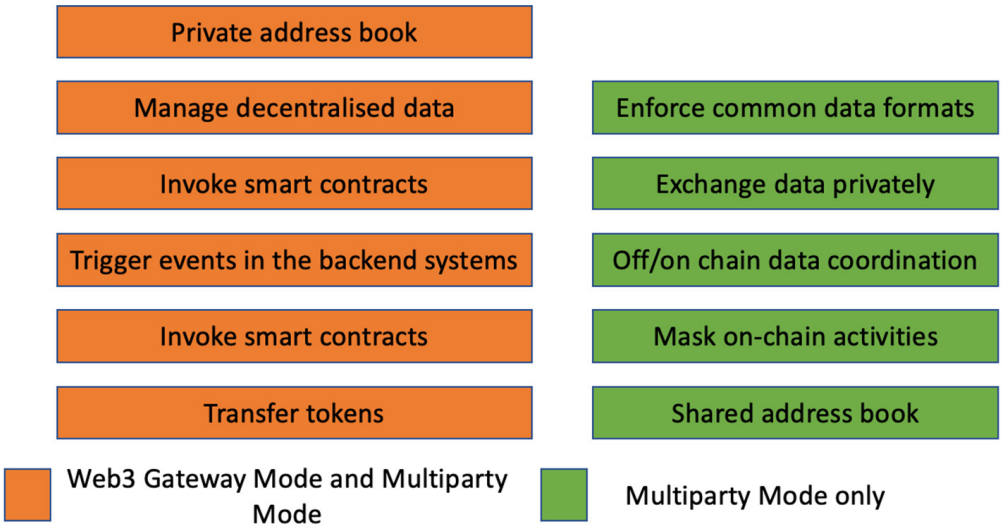

Figure 10. A Comparison Between Firefly Modes.

Firefly presents a noteworthy set of tools for bringing Blockchain technology to the AECO industry, as it shortens the steps required to build Blockchain applications. Microservices architecture allows it to be customised. Firefly Core, the main service in Firefly, contains the orchestration engine. The orchestration engine is a runtime environment that keeps the data and Blockchain state updated and communicates with the participating Blockchains and the public and private storage services through an event bus. The event bus is a secured data communication channel. Firefly has developer tools such as the Software Development Kit (SDK), APIs, and the Command-Line Interface (CLI). The CLI can create an offline Blockchain environment to facilitate the development process. Other tools include the Firefly Explorer, which is a part of the Core services and allows for system state monitoring, and the Firefly Sandbox, an external Firefly component that exists logically outside Firefly but acts as the end-user interface, which allows the user to communicate with Firefly through the Firefly API. The Sandbox is an example of the application frontends that can be developed with Firefly.

Firefly is a toolbox for developing Blockchain solutions. Depending on the purpose of the solution, the main tools can be categorised into the following groups: (1) Blockchain connectors: Firefly allows developers to connect to one or more Blockchains using connectors. Firefly supports several Blockchains out of the box, such as Ethereum and EVM chains, Hyperledger Fabric and Besu, and Quorum, with the possibility to build custom Blockchain connectors for other Blockchains. (2) Storage connectors allow Firefly-based applications to connect to storage infrastructures such as IPFS public and private nodes. (3) Indexing databases allowing Firefly nodes to index Blockchain blocks data for rich query purposes. (4) Data exchanging databases allow Firefly application users to communicate in an immutable way where each correspondence is linked to a unique hash stored in the Blockchain. (5) Token drivers are predefined tools used to facilitate performing operations including minting, transferring and burning on standardised tokens, such as ERC-20 and ERC-721. (6) Event listeners are integration tools that include Webhooks, WebSockets and system listeners. These can be configured to monitor certain digital activities coming internally from the system or outside sources. (7) APIs for smart contracts where Firefly could generate an API for any smart contract if the developer provides a FireFly Interface (FFI) format for the desired smart contract.

3.2.3. Application Workflow

Previously, tokens were discussed as a wide-spectrum tool to represent AECO assets; however, for this application, the tokens are used to represent building models. Two aspects should be clear when tokens represent assets: (1) actors and (2) token lifecycle. Each token

that represents an asset has a lifecycle created based on the original asset lifecycle. Actors are the lifecycle participants. Figure 11 below shows a lifecycle example for a building model with three actors: (1) the Architect who designed the model, (2) the civil engineer who oversees the model to satisfy the structural requirements, and (3) the consultant who ensures the design satisfies the owner's needs.

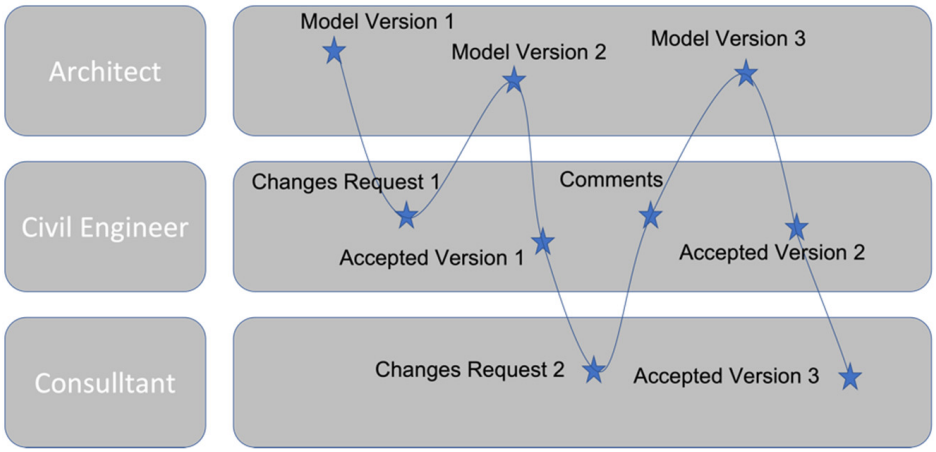

Figure 11. AECO NFT Lifecycle Example.

Based on the example, the asset lifecycle will start with a primary token issued by the Architect and secondary tokens issued by all actors. The secondary tokens present an activity related to the asset presented in the primary token. Therefore, each actor should create an ERC-721 smart contract to issue tokens. Figure 12 below is a snippet of the smart contract designed to issue tokens from the Architect's wallet.

```
// SPDX-License-Identifier: MIT
pragma solidity ^0.8.9;

import "@openzeppelin/contracts@4.8.0/token/ERC721/ERC721.sol";
import "@openzeppelin/contracts@4.8.0/token/ERC721/extensions/ERC721URIStorage.sol'
import "@openzeppelin/contracts@4.8.0/access/Ownable.sol";

contract DesignAssets is ERC721, ERC721URIStorage, Ownable {
    constructor() ERC721("DesignAssets", "DAST") {}

    function safeMint(address to, uint256 tokenId, string memory uri)
        public
        onlyOwner
    {
        _safeMint(to, tokenId);
        _setTokenURI(tokenId, uri);
    }

    // The following functions are overrides required by Solidity.

    function _burn(uint256 tokenId) internal override(ERC721, ERC721URIStorage) {
        super._burn(tokenId);
    }

    function tokenURI(uint256 tokenId)
        public
        view
        override(ERC721, ERC721URIStorage)
        returns (string memory)
    {
        return super.tokenURI(tokenId);
    }
}
```

Figure 12. ERC-721 Compatible Smart Contract.

The smart contract uses the Open zeppelin template written in Solidity to issue ERC-721 tokens. Three customisations for the template were made: (1) added a relevant contract name: "ProjectXDesigns"; (2) token name: "DesignAssets"; and (3) token symbol: "DAST".

The ERC-721 standard specifies that each NFT can have added metadata through a Uniform Resource Identifier (URI). The metadata of ERC-721 NFTs are unique in their on-chain identity. Further information can be added through a URI stored on-chain pointing at an off-chain file. The common practice is using the URI to direct to a JSON serialised file containing a detailed NFT attributes record. All files, including the JSON file, are stored in an IPFS distributed files storage system to avoid data centralisation. The JSON schema for this example is shown in Figure 13.


```
{
"name": "Building Architectural Model Project A Building Architectural Model",
"description": "This is the primary NFT for the ",
"IPNS": "ipns://k51qzi5uqu5dhcfir5hm5ks0g2fdrfnk4lqieduxopnzu7p7gor9wtasb6d8u8",
"IPFS": "ipfs://Qmcf2NR34SeXgAgumAjiXmTw9daJ8dgMfTk654w4qk7FFU",
"type": "Primary",
"Project": "Project A",
"attributes": [
{
"key": "Author",
"trait_type": "Author",
"value": "Architect X."
},
{
"key": "Version",
"trait_type": "Version",
"value": "1.0"
},
{
"key": "Type of Communication",
"trait_type": "Type of Communication",
"value": "New Version."
},
{
"key": "File Type",
"trait_type": "File Type",
"value": "IFC"
}
]
}
```


Figure 13. JSON Schema Designed to store Design Assets Information.

The JSON schema leverages IPFS Content addressing (CID) and InterPlanetary Name System (IPNS) systems. The CID creates a unique identifier for each file stored in the IPFS system [49]. If the content inside the file changes, the CID will change, which acts as an integrity indicator. IPNS is a pointing system that points at a file or a folder in IPFS and does not change if the content changes. Table 2 below explains the purpose of the NFT attributes shown in the JSON schema.

This proposed application has two types of activities: on-chain and off-chain. On-chain includes deploying the smart contract, minting an NFT, and adding NFT URI. Off-chain activities include indexing the NFT assets information, as shown in Figure 14.

Table 2. NFT Attributes Explained.

Attribute	Purpose
Name	Name of the asset
IPNS	Points to the project folder
IPFS	CID for the Design related file
NFT type	Primary or Secondary design asset
Project	The project connected to the NFT
Author	Author of the NFT design asset
Version	The assigned version for the NFT
Type of Communication	The theme of NFT (new version, change request, RFI)
File Type	The file type of the shared asset

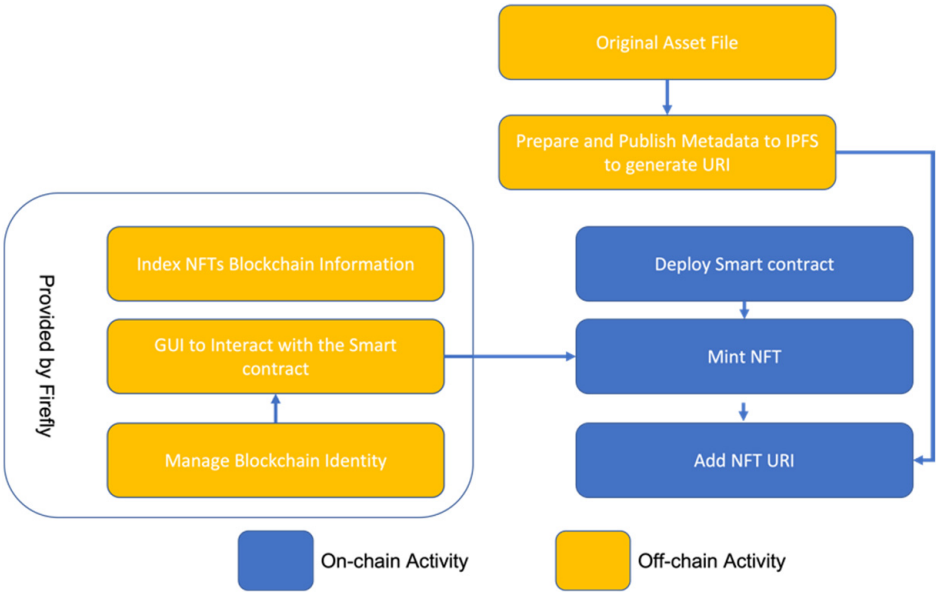

Figure 14. On-chain/Off-chain Activities Classification.

3.2.4. Working with the Firefly CLI

The Firefly CLI is a development environment for Blockchain applications. A Blockchain network was created using Firefly. The network consists of three hypothetical members: an Architect from the Design office, an Engineer from the construction site and a Consultant representing the owner. Figure 15 below shows the defined members in the Blockchain explorer, one of the Firefly services.

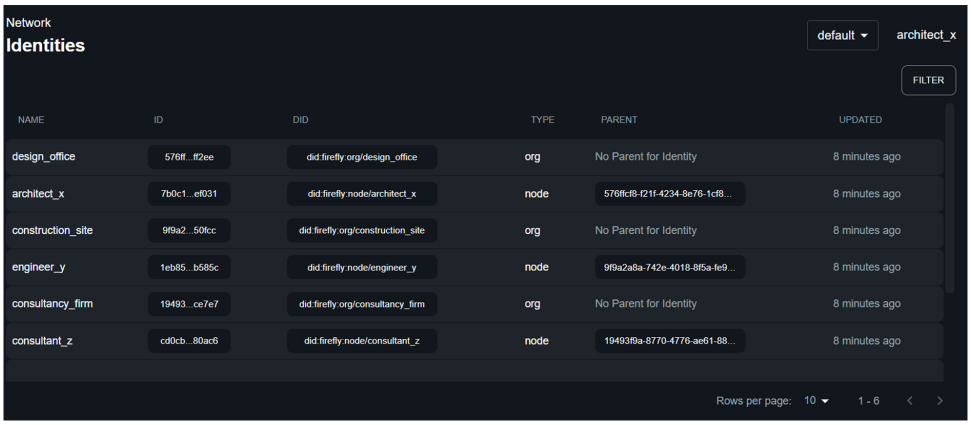

Figure 15. Defined Identities for the Application.

The CLI generates Docker environment files to simulate Blockchain and web services. These services include a local Blockchain, a web interface called Sandbox and an API generator.

- Customising the Firefly Local Environment

Firefly is an orchestrating tool between different components. The Firefly node can be customised to fit specific needs. Depending on the deployer's needs, these components can be customised by altering the config file of Firefly. For the purpose of this article, the namespace configuration will be altered to create a custom namespace called project A, consisting of three members: (1) Architect; (2) Engineer; and (3) Consultant. Another namespace will be created for illustration purposes called project B, consisting of one member: Architect. Figure 16 below shows the added configuration for the Architect environment config file to define the project A namespace.

```
namespaces:
  default: project_a
  predefined:
  - name: project_a
    defaultkey: 0x54dc1d1f22ecc58d0990f53483d4c151d89559b3
    description: project A members namespace
    plugins: [database0, blockchain0, sharedstorage0, erc20_erc721]
```

Figure 16. Project A Namespace Configuration.

The namespace requires a name and a default key representing the identity accessing the namespace. The plugins section defines the components of the namespace. The namespace uses database0 to index Blockchain data off-chain; blockchain0, an Ethereum-compatible local blockchain; sharedstorage0, a local IPFS node used to store files; and erc20_erc721, a token driver used to perform operations on standardised Ethereum tokens. Figure 17 shows the Architect node on namespace project A. Figure 18 shows the ability of the Architect to move between namespaces from the corner of the user interface page.

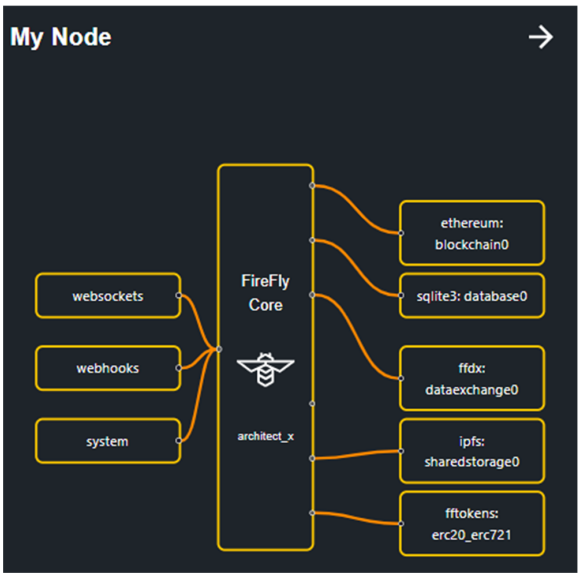

Figure 17. The Architect Node Components.

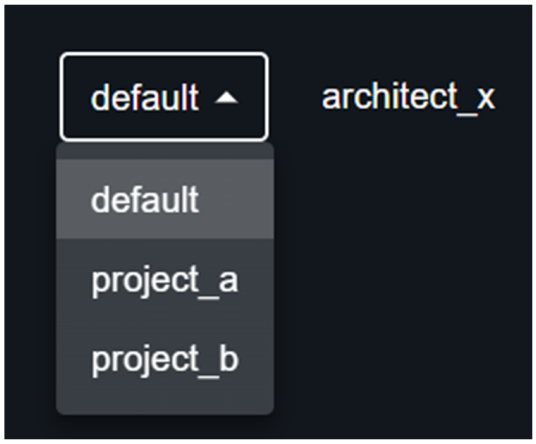

Figure 18. The Architect Namespaces.

3.2.5. Managing Design Assets with Firefly

The Firefly environment can be used in two ways to help manage assets with Blockchain: (1) using Firefly's basic smart contract; (2) using a custom contract. Firefly's basic smart contract is deployed on the Blockchain whenever Firefly is used. It allows Firefly to perform unique operations, such as messaging between the namespace members. Messaging is a sandbox feature that allows members to communicate off-chain with on-chain protection, where each message is hashed, and the hash is transacted to the Blockchain, which can be used later to verify the message's authenticity. The Sandbox allows for multiple message types such as text, defined data type or a file. Figure 19 below shows the messages block in the sandbox interface. This messaging can be used to exchange assets with the protection of Blockchain.

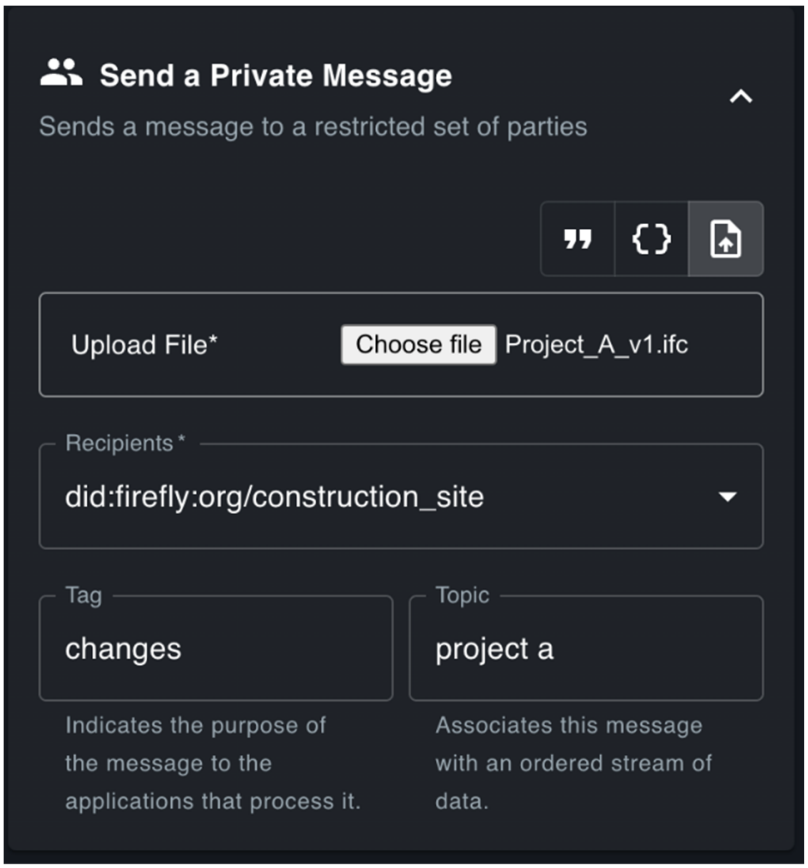

Figure 19. Messaging Among Project A Members through Firefly Sandbox.

Firefly can deal with custom contracts regardless of their function if they provide an interface file. Smart contracts are stored on the Blockchain in a bytecode format which is a machine-level code. Therefore, in order to make the smart contract understood by applications, they need to provide an interface file that acts as a dictionary for the smart contract. Firefly accepts two formats of smart contract interfaces: (1) Firefly Interface (FFI) [50] and (2) Application Binary Interface (ABI) [51]. The interface file provides a list of the smart contract functions, methods and events. FFI is a Firefly-specific file, while ABI is an Ethereum format used by EVM-compatible Blockchains. In order to use the smart contract provided within Firefly, the following steps are followed: (1) compile and deploy the smart contract on the Blockchain. The Ethereum Remix IDE [52] will be used for this purpose. Remix IDE will compile the Solidity file of the smart contract and generate the ABI interface file; (2) define the contract interface by adding the ABI file generated by Remix using the Sandbox: (3) generate an API for the defined smart contract. This API transforms the smart contract into an API that any application can use. Figure 20 shows the Sandbox steps.

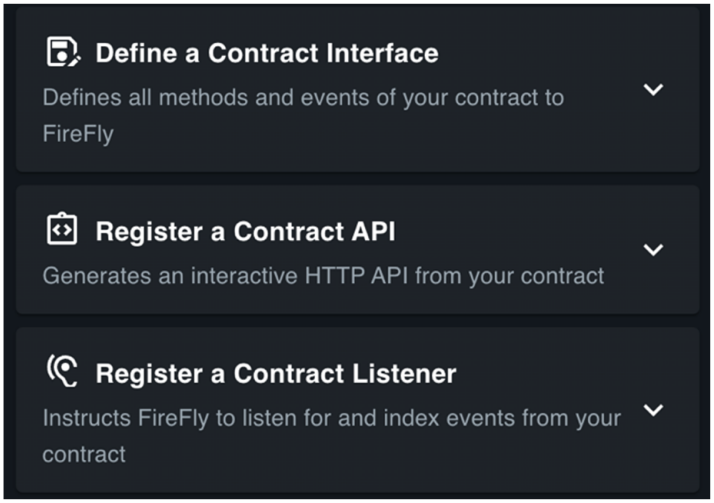

Figure 20. Smart Contract Recognition Through Firefly Sandbox.

The generated API shows the smart contract functions, methods and events as API operations. Figure 21 shows examples of such operations and functions.

Figure 21. Smart Contract API.

The function used to mint new NFTs in this smart contract is safeMint [45]. This function allows the deployer to mint NFTs that follow the ERC-721 standard. The function takes three parameters: "id", "to", and "URI". "id" is a unique identifier for the NFT within the smart contract. The "to" parameter takes the owner of the NFT address. The "URI" parameter links the token to a metadata file stored off-chain. Figure 22 below shows the request body for the safeMint function API operations where the smart contract controlled by the Architect can mint Designs modifications through this API.

Figure 22. Request Body for Using the safeMint function.

When this request is posted to the Blockchain through the API, an NFT will be minted with the details provided in the body. In the Firefly Blockchain explorer, the NFT-related activities can be seen. Figure 23 shows the minted NFT-related transactions.

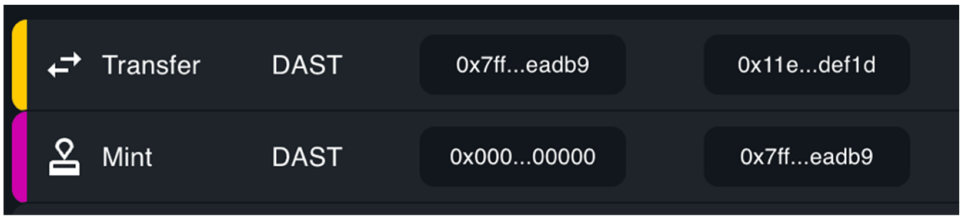

Figure 23. NFT-related event on the Blockchain Explorer.

4. Discussion and Conclusions

The article explores the fundamentals of developing Blockchain applications and customises them to fit the AECO industry. Blockchain concepts such as non-fungible tokens (NFTs), fungible tokens, namespaces and construction oracles were introduced. A novel approach to managing AECO assets using Blockchain technology was presented. Fundamental Blockchain elements such as smart contracts and oracles were discussed. The presented reference model to AECO Blockchain infrastructure rationalises the building components for Blockchain applications when applied to construction information systems. A framework for developing an AECO Blockchain application was introduced. The framework aims to streamline building Blockchain applications through six steps. The framework was applied to a demo application using Hyperledger Firefly to exchange non-fungible tokens (NFTs) representing AECO assets. During the process, Hyperledger Firefly concepts and features were explored, such as the API generator, which enabled the user to interact with the smart contract without difficulty.

Blockchain technology usage in the AECO industry is still limited as more studies appear to discuss its benefit and the possible methods of implementation [53]. However, this study tries to find a generic blueprint to develop Blockchain applications for the AECO industry, followed by an example following that blueprint. The blueprint discusses the

basic components of Blockchain applications based on the application purpose, followed by a discussion of the component's capacities. Furthermore, the article discusses NFTs as a tool for managing AECO assets and presents an orchestration tool for Blockchain applications, including NFTs. Firefly was used to generate identities, namespaces and smart contract APIs. Firefly is still in its early stages, as version one was released in April 2022, and further time is required to achieve maturity. However, it is a promising tool for building Blockchain applications quickly and efficiently.

Further future research can contribute to this work by customising the framework to a specific field of construction operations, such as construction supply chains or stockholder collaboration. Blockchain adoption in the AECO sector is still incipient. As technological maturity increases, the acceptability of the proposed framework by the industry must be validated beyond literature reviews and proof-of-concept studies. As such, planned future developments include undertaking surveys with industry stakeholders and performing case studies in an operational environment.

Author Contributions: Conceptualization, M.D.; methodology, M.D.; writing—original draft preparation, M.D.; writing—review and editing, M.D. and J.P.M.; supervision, J.P.M. All authors have read and agreed to the published version of the manuscript.

Funding: The authors acknowledge the financial support from the Portuguese Foundation for Science and Technology (FCT) through the PhD Grant 2020.05786. BD. This work was financially supported by Base Funding–UIDB/04708/2020 of the CONSTRUCT–Instituto de I&D em Estruturas e Construções—funded by national funds through the FCT/MCTES (PIDDAC).

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Zhang, Y.; Wang, T.; Yuen, K.V. Construction site information decentralized management using blockchain and smart contracts. *Comput. AIDED Civ. Infrastruct. Eng.* **2022**, *37*, 1450–1467. [CrossRef]
2. Penzes, B.; KirNup, A.; Gage, C.; Dravai, T.; Colmer, M. *Blockchain Technology in the Construction Industry: Digital Transformation for High Productivity*; Institution of Civil Engineers (ICE): London, UK, 2018.
3. Adams, H.; Zinsmeister, N.; Salem, M.; Keefer, R.; Robinson, D. Uniswap v3 core. *Tech. Rep. Uniswap Tech. Rep.* **2021**.
4. White, B.; Mahanti, A.; Passi, K. Characterizing the OpenSea NFT Marketplace. In Proceedings of the Web Conference 2022, New York, NY, USA, 25 April 2022; pp. 488–496. [CrossRef]
5. Breidenbach, L.; Cachin, C.; Chan, B.; Coventry, A.; Ellis, S.; Juels, A.; Koushanfar, F.; Miller, A.; Magauran, B.; Moroz, D.; et al. Chainlink 2.0: Next steps in the evolution of decentralized oracle networks. *Chainlink Labs* **2021**, *1*.
6. Tal, Y.; Ramirez, B.; Pohlmann, J. The Graph: A Decentralized Query Protocol for Blockchains. 2018. Available online: https://raw.githubusercontent.com/graphprotocol/research/master/papers/whitepaper/the-graph-whitepaper.pdf (accessed on 14 December 2022).
7. Gudgeon, L.; Werner, S.; Perez, D.; Knottenbelt, W.J. DeFi Protocols for Loanable Funds: Interest Rates, Liquidity and Market Efficiency. In Proceedings of the 2nd ACM Conference on Advances in Financial Technologies, New York, NY, USA, 21–23 October 2020; pp. 92–112. [CrossRef]
8. Curve.fi, Curve Documentation. Release 1.0.0. 2022. Available online: https://curve.readthedocs.io/_/downloads/en/latest/pdf/ (accessed on 14 December 2022).
9. Goanta, C. Selling LAND in Decentraland: The Regime of Non-fungible Tokens on the Ethereum Blockchain Under the Digital Content Directive. In *Disruptive Technology, Legal Innovation, and the Future of Real Estate*; Lehavi, A., Levine-Schnur, R., Eds.; Springer International Publishing: Cham, Switzerland, 2020; pp. 139–154. [CrossRef]
10. Dowling, M. Is non-fungible token pricing driven by cryptocurrencies? *Finance Res. Lett.* **2022**, *44*, 102097. [CrossRef]
11. Novak, K. Introducing the Metaverse, Again! *TechTrends* **2022**, *66*, 737–739. [CrossRef]
12. Rajendran, B.; Palaniappan, G.; Dijesh, R.; Bindhumadhava Bapu, S.; Sudarsan, S.D. A Universal Domain Name Resolution Service—Need and Challenges—Study on Blockchain Based Naming Services. In Proceedings of the 2022 IEEE Region 10 Symposium (TENSYMP), Mumbai, India, 1–3 July 2022; pp. 1–6. [CrossRef]
13. Basu, S.; Fernald, J. Information and communications technology as a general-purpose technology: Evidence from US industry data. *Ger. Econ. Rev.* **2007**, *8*, 146–173. [CrossRef]
14. Valavanis, K.P.; Vachtsevanos, G.J.; Antsaklis, P.J. Technology and autonomous mechanisms in the mediterranean: From ancient Greece to Byzantium. In Proceedings of the 2007 European Control Conference (ECC), 2 July 2007; pp. 263–270. [CrossRef]
15. Wang, Z.; Wang, T.; Hu, H.; Gong, J.; Ren, X.; Xiao, Q. Blockchain-based framework for improving supply chain traceability and information sharing in precast construction. *Autom. Constr.* **2020**, *111*, 103063. [CrossRef]

16. Yevu, S.K.; Yu, A.T.W.; Darko, A. Digitalization of construction supply chain and procurement in the built environment: Emerging technologies and opportunities for sustainable processes. *J. Clean. Prod.* **2021**, *322*, 129093. [CrossRef]
17. Hijazi, A.A.; Perera, S.; Calheiros, R.N.; Alashwal, A. A data model for integrating BIM and blockchain to enable a single source of truth for the construction supply chain data delivery. *Eng. Constr. Archit. Manag.* **2022**. *Epub ahead of printing*. [CrossRef]
18. Ciotta, V.; Mariniello, G.; Asprone, D.; Botta, A.; Manfredi, G. Integration of blockchains and smart contracts into construction information flows: Proof-of-concept. *Autom. Constr.* **2021**, *132*, 103925. [CrossRef]
19. Pradeep, A.S.E.; Yiu, T.W.; Zou, Y.; Amor, R. Blockchain-aided information exchange records for design liability control and improved security. *Autom. Constr.* **2021**, *126*, 103667. [CrossRef]
20. Lou, J.; Lu, W. Construction information authentication and integrity using blockchain-oriented watermarking techniques. *Autom. Constr.* **2022**, *143*, 104570. [CrossRef]
21. Xue, F.; Lu, W.S. A semantic differential transaction approach to minimizing information redundancy for BIM and blockchain integration. *Autom. Constr.* **2020**, *118*, 103270. [CrossRef]
22. Rane, S.B.; Narvel, Y.A.M. Data-driven decision making with Blockchain-IoT integrated architecture: A project resource management agility perspective of industry 4.0. *Int. J. Syst. Assur. Eng. Manag.* **2022**, *13*, 1005–1023. [CrossRef]
23. Elghaish, F.; Matarneh, S.T.; Edwards, D.J.; Rahimian, F.P.; El-Gohary, H.; Ejohwomu, O. Applications of Industry 4.0 digital technologies towards a construction circular economy: Gap analysis and conceptual framework. *Constr. Innov.* **2022**, *22*, 647–670. [CrossRef]
24. Balasubramanian, S.; Shukla, V.; Islam, N.; Manghat, S. Construction Industry 4.0 and Sustainability: An Enabling Framework. *IEEE Trans. Eng. Manag.* **2022**, 1–19. [CrossRef]
25. Parn, E.A.; Edwards, D. Cyber threats confronting the digital built environment: Common data environment vulnerabilities and block chain deterrence. *Eng. Constr. Archit. Manag.* **2019**, *26*, 245–266. [CrossRef]
26. de Soto, B.G.; Georgescu, A.; Mantha, B.; Turk, Z.; Maciel, A.; Semih, M. Construction cybersecurity and critical infrastructure protection: New horizons for construction 4.0. *J. Inf. Technol. Constr.* **2022**, *27*, 571. [CrossRef]
27. Pargoo, N.S.; Ilbeigi, M. A Scoping Review for Cybersecurity in the Construction Industry. *J. Manag. Eng.* **2023**, *39*, 03122003. [CrossRef]
28. Yang, R.; Wakefield, R.; Lyu, S.; Jayasuriya, S.; Han, F.; Yi, X.; Yang, X.; Amarasinghe, G.; Chen, S. Public and private blockchain in construction business process and information integration. *Autom. Constr.* **2020**, *118*, 103276. [CrossRef]
29. De Lauretis, L. From Monolithic Architecture to Microservices Architecture. In Proceedings of the 2019 IEEE International Symposium on Software Reliability Engineering Workshops (ISSREW), Berlin, Germany, 27–30 October 2019; pp. 93–96. [CrossRef]
30. Bucchiarone, A.; Dragoni, N.; Dustdar, S.; Larsen, S.T.; Mazzara, M. From Monolithic to Microservices: An Experience Report from the Banking Domain. *IEEE Softw.* **2018**, *35*, 50–55. [CrossRef]
31. Albert, E.; Gordillo, P.; Livshits, B.; Rubio, A.; Sergey, I. EthIR: A Framework for High-Level Analysis of Ethereum Bytecode. In *Automated Technology for Verification and Analysis*; Springer International Publishing: Cham, Switzerland, 2018; pp. 513–520. [CrossRef]
32. Kapengut, E.; Mizrach, B. An Event Study of the Ethereum Transition to Proof-of-Stake. *arXiv* **2022**, arXiv:2210.13655. [CrossRef]
33. Beniiche, A. A Study of Blockchain Oracles. *arXiv* **2020**, arXiv:2004.07140. [CrossRef]
34. Mühlberger, R.; Bachhofner, S.; Ferrer, E.C.; Di Ciccio, C.; Weber, I.; Wöhrer, M.; Zdun, U. Foundational Oracle Patterns: Connecting Blockchain to the Off-Chain World. In *Business Process Management: Blockchain and Robotic Process Automation Forum*; Springer International Publishing: Cham, Switzerland, 2020; pp. 35–51. [CrossRef]
35. Zhang, Z.; Karamanolis, C. Designing a robust namespace for distributed file services. In Proceedings of the 20th IEEE Symposium on Reliable Distributed Systems, New Orleans, LA, USA, 28–31 October 2001; pp. 162–171. [CrossRef]
36. Alharby, M.; van Moorsel, A. Blockchain-based Smart Contracts: A Systematic Mapping Study. In *Computer Science & Information Technology (CS & IT)*; AIRCC: Chennai, India, 2017; pp. 125–140. [CrossRef]
37. Bandara, E.; Liang, X.; Foytik, P.; Shetty, S.; Ranasinghe, N.; De Zoysa, K. Rahasak—Scalable blockchain architecture for enterprise applications. *J. Syst. Archit.* **2021**, *116*, 102061. [CrossRef]
38. The Ordering Service—Hyperledger-Fabricdocs Main Documentation. Available online: https://hyperledger-fabric.readthedocs.io/en/release-2.5/orderer/ordering_service.html (accessed on 14 December 2022).
39. Lee, D.; Lee, S.H.; Masoud, N.; Krishnan, M.S.; Li, V.C. Integrated digital twin and blockchain framework to support accountable information sharing in construction projects. *Autom. Constr.* **2021**, *127*, 103688. [CrossRef]
40. What Is an Oracle in Blockchain? Explained | Chainlink. Available online: https://chain.link/education/blockchain-oracles (accessed on 14 December 2022).
41. ISO. ISO 19650-1:2018. 2018. Available online: https://www.iso.org/cms/render/live/en/sites/isoorg/contents/data/standard/06/80/68078.html (accessed on 30 March 2022).
42. Bamakan, S.M.H.; Nezhadsistani, N.; Bodaghi, O.; Qu, Q. Patents and intellectual property assets as non-fungible tokens; key technologies and challenges. *Sci. Rep.* **2022**, *12*, 1–3. [CrossRef]
43. Kugler, L. Non-fungible tokens and the future of art. *Commun. ACM* **2021**, *64*, 19–20. [CrossRef]
44. Entriken, W.; Shirley, D.; Evans, J.; Sachs, N. EIP-721: Non-Fungible Token Standard. Ethereum Improvement Proposals. 2018. Available online: https://eips.ethereum.org/EIPS/eip-721 (accessed on 13 April 2022).

45. ERC 721—OpenZeppelin Docs. Available online: https://docs.openzeppelin.com/contracts/4.x/api/token/ERC721 (accessed on 14 December 2022).
46. Introduction to Hyperledger FireFly. Hyperledger FireFly Docs. Available online: https://hyperledger.github.io/firefly/overview/supernode_concept.html (accessed on 14 December 2022).
47. Voshmgir, S. *Token Economy: How the Web3 Reinvents the Internet*; Token Kitchen: Berlin, Germany, 2020.
48. Usage Patterns. Hyperledger FireFly Docs. Available online: https://hyperledger.github.io/firefly/overview/usage_patterns.html (accessed on 14 December 2022).
49. Benet, J. Ipfs-content addressed, versioned, p2p file system. *arXiv* **2014**, arXiv:1407.3561.
50. FireFly Interface Format. Hyperledger FireFly Docs. Available online: https://hyperledger.github.io/firefly/reference/firefly_interface_format.html (accessed on 14 December 2022).
51. What Is an ABI? Explained—Step-by-Step Beginners Guides | QuickNode. Available online: https://www.quicknode.com/guides/smart-contract-development/what-is-an-abi (accessed on 14 December 2022).
52. Solidity Compiler—Remix—Ethereum IDE 1 Documentation. Available online: https://remix-ide.readthedocs.io/en/latest/compile.html (accessed on 14 December 2022).
53. Darabseh, M.; Martins, J.P. Risks and opportunities for reforming construction with blockchain: Bibliometric study. *Civ. Eng. J. Iran* **2020**, *6*, 1204–1217. [CrossRef]