COGITO

CONSTRUCTION PHASE DIGITAL TWIN MODEL

cogito-project.eu

D6.2 – Blockchain & Smart Contracts on the Workflow Modelling and Management v2

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 958310

D6.2 – Blockchain & Smart Contracts on the Workflow Modelling and Management v2

Dissemination Level:	Public
Deliverable Type:	Demonstrator
Lead Partner:	QUE
Contributing Partners:	Hypertech, UCL, BOC-AG, NT
Due date:	31-10-2022
Actual submission date:	31-10-2022

Authors

Name	Beneficiary	Email
Panagiotis Andriopoulos	QUE	panos@que-tech.com
Panagiotis Moraitis	QUE	p.moraitis@que-tech.com
Giorgos Koletsas	QUE	g.koletsas@que-tech.com
Stefanos-Dimitris Kakavoulis	QUE	s.d.kakavoulis@que-tech.com
Anastasia Fotopoulou	Hypertech	a.fotopoulou@hypertech.gr
Theofania Charbi	Hypertech	t.charbi@hypertech.gr
Evangelos Karalis	Hypertech	e.karalis@hypertech.gr
Bohuš Belej	NT	belej@novitechgroup.sk
Martin Straka	NT	straka@novitechgroup.sk

Reviewers

Name	Beneficiary	Email
Tobias Hanel	FER	thanel@ferrovial.com
Thanos Tsakiris	CERTH	atsakir@iti.gr

Version History

Version	Editors	Date	Comment
0.1	QUE	10.05.2022	ToC and work allocation
0.2	QUE	02.07.2022	Updates on BC technology
0.3	QUE	28.08.2022	API Services
0.4	QUE	12.09.2022	Usage Walkthrough
0.5	QUE	21.10.2022	First Draft for internal review
0.6	FER, CERTH	27.10.2022	Internal review
0.7	QUE	27.10.2022	Internal review comments addressed
0.8	QUE	31.10.2022	Final version
1.0	QUE, Hypertech	31.10.2022	Submission to the EC portal

Disclaimer

Executive Summary

The complex nature of construction projects requires mutual trust and close cooperation between a large number of stakeholders, that more than often represent different interests. In this highly fragmented environment where unobscured cooperation is at constant risk, organization relies heavily on bureaucracy, conflicts and payment delays are common, the implementation of a trustless task completion verification system is a necessity. The solution that is presented here relies on Smart Contract and Blockchain technology to transform the construction industry into a more efficient and transparent environment. It creates a data and transactions governing tool to confirm the completion of tasks and works alongside with already existing methodologies. Through a collection of data from various actors it calculates a set of KPIs to evaluate the progress of construction tasks. This will eliminate the need of a central authority to govern data, and therefore it will create an environment of mutually beneficial transactions.

Table of contents

List of Figures

List of Tables

List of Acronyms

Term	Description
ABCI	Application Blockchain Interface
COGITO	Construction Phase diGItal Twin mOdel
BC-SC	Blockchain - Smart Contract
BFT	Byzantine Fault Tolerant
BIM	Building Information Modelling
DLT	Distributed Ledger Technology
EVM	Ethereum Virtual Machine
IBC	Inter Blockchain Communication
KPIs	Key Performance Indicators
P2P	Peer-to-Peer
PoS	Proof of Stake
PoW	Proof of Work
SC	Smart Contract
SLA	Service Level Agreement
SLAM	Service Level Agreement Manager
SLO	Service Level Objective
SMEs	Small and Medium Enterprises
WODM	Work Order Execution Assistance
WOEA	Work Order Execution Assistance

1 Introduction

1.1 Scope and Objectives of the Deliverable

The main objective of the document is to give an overview of the Smart Contract technology as the tool to increase the trustworthiness of work order verification, and how this can become an efficient, flexible and scalable solution. To this purpose, it describes in depth current applications and studies on Blockchain and Smart Contract technologies, their potential and their bottlenecks. Based on these findings, it describes in detail how the current solution will be seamlessly integrated in COGITO project, in order to enhance and automate the current workflow. This approach eliminates the need of a central authority to govern data related to task progression, and therefore it creates an environment of mutually trusted exchange of data between the various stakeholders. A number of construction-related KPIs are proposed, that enhance the potential of Smart Contract enabled Service Level Agreements on automation, security, immutability, and decentralization of contractual transactions. A detailed description of the proposed methodology is presented; how a conventional paper-based SLA, derived from a work order, is translated from a document to a smart piece of self-executing code. For that reason, the rules and processes that govern the relationships between one or more parties are transformed into a conceptual data model, with distinct entities, properties, and relationships.

This document also presents the technological framework is used for the implementation of the Smart Contract methodology. Two highly interconnected frameworks are used. The first one is responsible for the unobscured operation of the Blockchain layer and consists of the Cosmos SDK and the Tendermint Byzantine Fault Tolerant (BFT) engine, while the second one is dedicated to the business logic of Smart Contract functionality and consists of a well-tested and very efficient combination of Evmos and Solidity.

The final objective of D6.2 is to thoroughly present the services that will be offered through the dedicated APIs that will undertake the communication between the Smart Contract platform, the Blockchain plugin and the Work Order Definition Monitoring tool (WODM – see deliverable D2.4 – COGITO System Architecture v1), alongside the common data model of the exchanged information.

1.2 Relation to other Tasks and Deliverables

For this document valuable inputs have been received from tasks T2.1 – “Elicitation of Stakeholder Requirements”, T2.4 – “COGITO System Architecture Design”, and the associated deliverables D2.1 – “Stakeholder requirement for the COGITO system” and D2.4 – “COGITO System Architecture v1”. Additionally, there was a close cooperation and exchange of inputs and information with T3.1 – “Survey of Existing Data Models & Ontologies & Associated Standardization Efforts” and T3.2 – “COGITO Data Model, Ontology Definition and Interoperability Design”. Finally, the current deliverable was directly connected with T6.3 – “Adaptive Workflow Management and Automation”. T6.3 is responsible for providing both the construction and the process model and based on that to present the interaction between different tools, activities and stakeholders. Based on these outcomes, the Smart Contract functionality presented in T6.1 will add a layer of security and mutual trust between the interacting parties.

1.3 Updates to the 1st release

The present document continuous on the foundations that were set from D6.1. While the first version presented the theoretical background of the solution alongside with a simple demonstrator, the present one delivers a fully developed tool consisted of two components. The KPIs that monitor the progress of a task have been refined, the data model has been improved to become more well-structured and the APIs have been redesigned to fully integrate all the services that are required. Finally, a full-scale solution has been developed and tested and the results are presented in a new Usage Walkthrough section.

1.4 Structure of the Deliverable

In section 2 of the deliverable a detailed description of the operating principles behind the Blockchain in construction industry and the added benefits that could be achieved within the COGITO project are presented.

Additionally, the methodology of translating conventional work orders and SLAs into a fully functional Smart Contract framework is presented. Section 3 describes the technologies that are used for the implementation and development of the SLAM and BC-SC components, alongside with the API documentation. This section also delivers a thorough walkthrough to make the user familiar with the services offered by these components.

2 Blockchain in the Construction Industry

2.1 Blockchain Technology

Blockchain technology at its core is a record-keeping tool, and it can be characterized as a type of Distributed Ledger Technology (DLT) which offers a decentralized peer-to-peer (P2P) system of information exchange and storage. A DLT in simple terms, is an accounting system where the record of transactions, the digital ledger, instead of being centrally stored is distributed among a network of computers and servers, which are called nodes [1]. The data in DLT systems are distributed, stored and accessed using cryptographic keys, enabling the secure functioning of the decentralized database. The main idea behind Blockchain technology is illustrated in Figure 1. In the first case (Figure 1a) there is one central accounting book where all transactions between the contracting parties are kept, and in the second (Figure 1b) there is a Blockchain based system where multiple records are simultaneously maintained and updated by multiple users.

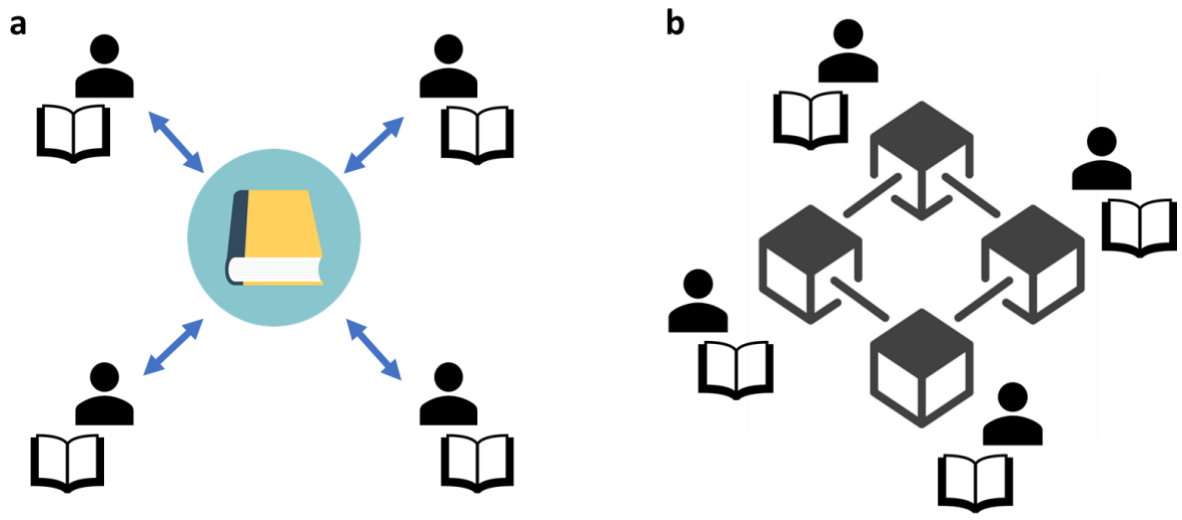


Figure 1 – Centralized (a) and Distributed record (b) keeping.

Blockchain Technology allows data that were conventionally held in a single database to be spread out among a network of nodes. This decentralization comes with many benefits besides the **redundancy** that guarantees the fidelity of the stored data. By removing the centralization of data storage, the need for a centralized authority to manage and control the data is also eliminated. This creates **transparency**, since every transaction is visible to the entire network, and no shady actions can take place. Moreover, it significantly reduces the vulnerability of the system, that now relies on multiple nodes and not on a single point that is sensitive to failures or malicious attacks. Since every transaction is encrypted and cryptographically linked to the previous transaction, it is far more **secure** than other methods of record-keeping. Once a transaction is confirmed and validated then it is impossible to change or to be deleted, leading to an **immutable** storage system. This permanent record of timestamped transactions can also be used to reliably and efficiently track information over time.

In a Blockchain all the information is anonymously recorded and cryptographically collected and shared in groups that are called blocks. This piece of information is in its most common form a set of transactions data, mainly due to the great publicity of cryptocurrencies, but the technology itself is not limited to this, and therefore a Blockchain can store any type of data. As it can be seen in Figure 2, the information is organized and stored in blocks. Every authenticated and authorized user of the Blockchain can submit a transaction, which is collected in a block alongside with the transactions of the other users. In order for a block to be valid, it has to be verified by the rest of the network using specific algorithms. These are called consensus mechanisms and the most popular are the Proof of Work (PoW) and the Proof of Stake (PoS), with the first one being heavily criticized for high energy consumption [2]. Once a block reaches its capacity, it is closed and linked to the previous block with a

hash. A hash is a unique sting of characters that is created by a special cryptographic function, and can be considered as the ID of each block.

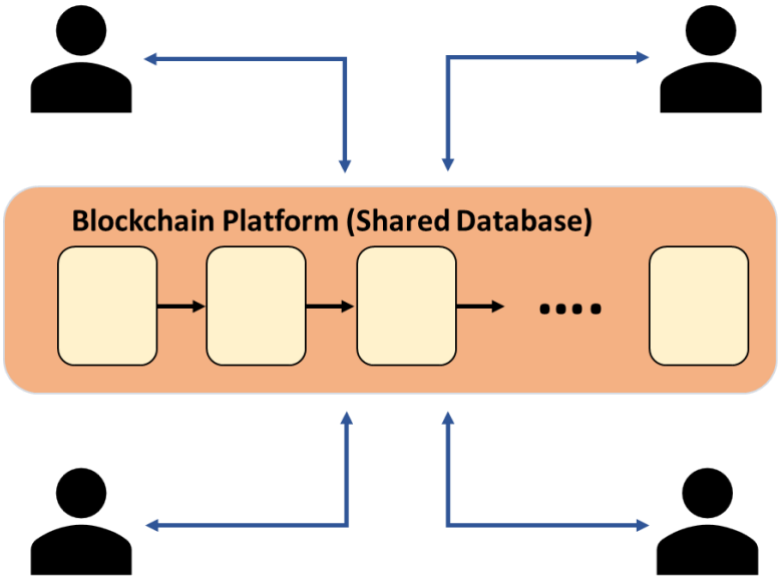


Figure 2 – Blockchain structure: Authenticated and authorized users can read and write data on the Blockchain.

2.2 Service Level Agreements and Smart Contracts

A Service Level Agreement (SLA) is a contract between a service provider and a customer that describes the level of the service that the customer expects to receive from the service provider. An SLA describes in clear, measurable and quantifiable ways the objectives of the contractual agreement between the parties, alongside with performance standards that are required for each service. Those performance standards are constantly monitored through a set of Key Performance Indicators (KPIs), and usually failing to meet the desired levels of performance triggers a penalty. On the contrary if the service provider exceeds those standards, a reward or a bonus might be included in the terms of an SLA. These types of contracts are essential for building trust between the involved parties, as they ensure that both are on the same page regarding expectations and responsibilities. The typical form of an SLA is a paper document, and the KPIs of the contractual agreement are either monitored from the service provider or a third party. The latter case increases the cost of the delivered service and the first might cause distrust or conflicts. However, DLT provide a solution that comes with minimum cost and can restore trust.

A technological innovation that is associated with the Blockchain Technology are the Smart Contracts (SCs). A SC is a piece of self-executing computer algorithm that automatically runs when predetermined conditions are met [3]. They are typically used to automate, verify and enforce the execution of a contractual agreement such as an SLA without the involvement of an intermediary party. They are deployed and executed on a blockchain, and their scope is to express in a tamper proof manner, the terms that are contained within a conventional contract. They are inseparable from the underlying Blockchain technology, that can be characterized as a base layer technology, and it is mainly concerned with issues such as distributed data storage, cryptographic security and reaching consensus between the nodes of the network.

With the development of SCs on top of the Blockchain structure, self-executing programs can make changes to the distributed ledger based on a set of predefined rules. Compared to conventional contractual agreements, SCs owe their “smart” nature to their inherent ability to self-enforce. Therefore, a solid, secure but rather inflexible database system is enriched with automation processes. At the same time conventional contractual transactions inherit the underlying features of the blockchain architecture to become secure, immutable and decentralized. Also, since they are part of a blockchain platform, they operate under the same principle of

simultaneous execution by many nodes, and therefore they are fully traceable and transparent, while they can enforce immediate transfer of funds if necessary. The necessity for a mediator is eliminated and no central authority or third party can dishonestly control the agreement.

However, since SCs are a form of an agreement between two or more parties, they share common features with their conventional paper-based siblings. This can also be depicted in Figure 3, that briefly shows the lifecycle of a SC through a 4-step process. During the first step, a customer identifies, selects and negotiates with a service provider regarding the delivery of a service (step 1). The service provider will provide the means to deliver the service, and this process while be evaluated according to the terms of the contract through a set of KPIs. Once the agreement is established the SC is deployed and shared among the blockchain nodes (step 2). The SC continuously monitors if the service is properly delivered or not by monitoring the KPIs of the agreement (Step 3). If the KPIs are within the contracted levels, then the whole process continues unobscured, otherwise the customer is reimbursed accordingly (Step 4). The transfer of funds is immediate, secure and transparent as it relies on encrypted information, and the execution runs simultaneously on multiple nodes instead of a single server.

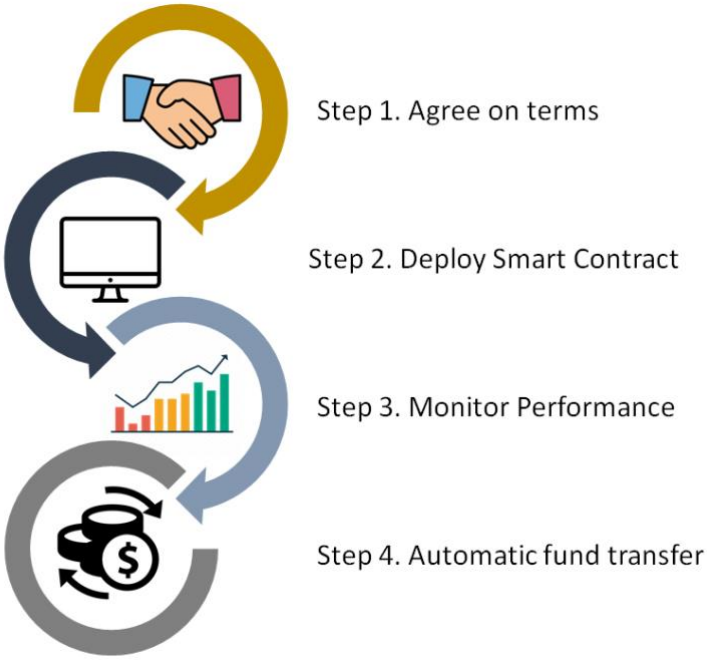


Figure 3 – Smart Contract lifecycle.

2.3 Blockchain and Smart Contracts in Construction Industry

With various industries such as Banking, Financial Services, Government, Healthcare and Supply Chain to have already started to adopt Blockchain technology, the construction industry is only at the beginning of such an attempt [4]. The construction industry can be described as a decentralized, loosely coupled network of individual stakeholders, and a notoriously slow industrial sector in the adoption of new technologies [5]. However, the application of Blockchain solutions, complemented with the development of the associated tools, might be especially promising, as it has the potential to address most of sector's key challenges [6].

The various construction projects are characterized of high complexity and limited repeatability. They are often delivered by a large number of teams that operate in a cross functional environment and are supported by a complex and fragmented supply chain system [7]. They consist of various composite tasks, and their successful completion relies on the seamless cooperation between teams and individuals in every step of the construction process. Therefore, the development of mutual trust between the interacting parties is more than necessary, not only for the completion of the project, but also to avoid delays and keep the construction within the planned budget [8]. However, the decentralized and project-based nature of construction projects requires many

stakeholders that represent different interests, and have different requirements, to interact and cooperate over long-time horizons. Due to the lack of trust, the numerous laws and regulations, and the short-term nature of the construction contracts, the latter are characterized by a complex contractual structure [9]. This combined with low digitization levels, slow process automation and lengthy inspection processes might result to persistent disputes and delayed payments. In this environment, where seamless cooperation is at constant risk and organization relies heavily on bureaucracy, conflicts and payment delays are common, the implementation of Blockchain technology can make construction more efficient and transparent.

The idea of using Blockchain in construction is not new, and a lot of research has been done due to its huge potential [10]. By introducing information traceability, transparency and most importantly, a single source of truth many applications in construction can be benefited. Blockchain by itself, as a baseline technology can increase trust to improve collaboration, the addition of smart contract functionality, can automate processes and it can eventually decrease bureaucracy, and finally, IoT devices can provide digitization of assets and processes allowing seamless integration and exchange of information between systems, components and actors. Potential use cases of Blockchain applications in construction according to literature can be clustered in 3 main categories [11]:

- **Document Management and Administration Purposes.** During the lifetime of a construction project a great volume of documents is produced as a result of various discontinuous processes such as design collaboration, work order management etc. More than often a document for the same process is constantly updated, resulting in many different versions that have to be carefully annotated and communicated to the consortium of the stakeholders. Blockchain and smart contracts could offer a solid framework for tracking workflows and document versioning, to facilitate document approval, validate authenticity, and cryptographically secure data confidentiality [12].
- **Transaction Automation.** Cashflow management is crucial for the construction industry, the wellbeing of the related technical and engineering companies, and the timely delivery of the projects within the planned budget. The greatest portion of payments is directed to the general contractor, subcontractors, supplies, equipment providers etc. However, delays are common and this causes conflicts, or even jeopardizes the existence of Small and Medium Enterprises (SMEs) that operate with marginal profits [13]. A SC solution could automate the issue of payments and also bring transparency and traceability. In a straightforward example, work hours on a construction site could be registered either manually or by utilizing inputs from various sensors and the payment could be issued automatically. Even on the planning phase, milestones can be bundled with smart contracts that are programmed to trigger payments when the deliverables are completed. In such a scenario, the payment could be initiated from the Blockchain, but to be linked with the projects bank account so that the transaction is made in fiat currency.
- **Immutable Record Keeping.** Transparency and immutability in record keeping is the key characteristic of DLTs and Blockchain can excel in this field, for meeting the requirements of the construction industry. It can provide a record of timestamped transactions between the involved parties, a record of changes in digital models, tracking of supply chain logistics including procurement, transportation, delivery and storage, a record of ownership of physical assets, or to keep a log with the personnel and the equipment on site [14].

The above categorization aims to present the fields that Blockchain technology and SCs can contribute to enhance the efficiency of construction industry. In the majority of use cases recorder in literature, features from more than one category are utilized to deliver a service. Of great importance is the contribution of **Building Information Modeling (BIM)** as it introduced a completely new way of working, connecting both the stakeholders and the data, not only during the planning and construction phase, but also across the entire building lifecycle. Bridging Blockchain and BIM can enable the creation of a new scheme of organizational structure that is devoid of hierarchy and centralized decision-making, while it addresses issues such as transaction speed, security and data integrity [15]. Blockchain could be an integrated part of BIM workflow during the planning phase, so that payments could be issued on time when specific tasks are completed. Additionally, it can sufficiently archive BIM changes and modification history, secure intellectual property rights, issue and annotate documents or invoices, organize and supervise the supply chain management, record health or safety hazards, and improve quality control or even waste management [16].

Despite the enormous potential, most scholars agree that Blockchain technology will not be adopted overnight and that a maturation period is necessary before it becomes an industry standard. Skepticism is prominent even to the most tech-savvy construction stakeholders to adopt Blockchain into their day-to-day operations. The non-traditional nature of blockchain and the unconventional way of processing transactions can be a tough challenge even for heavily digitized and innovative companies. Time, effort and investment capital would be necessary to adopt changes, train people and bridge the technological and cultural gap. However, small steps and careful planning could gradually unveil the potential while minimizing the risks, and this the approach that is followed in the COGITO project.

2.4 The Added Value of COGITO Solution

2.4.1 Overview

The goal of the present document is to design the SCs that will be used to enhance transparency and provide the trusted means to verify the completion of the construction tasks in COGITO project. The SCs dedicated to this goal will be based on the SLAs, that will be compatible with the workflow tasks. The workflow tasks, as a part of the work orders, will be derived from the **Work Order Definition and Modeling (WODM)** tool. WODM is the component responsible for creating work order templates, generate workflow tasks and monitor the execution of the defined workflow. The SCs functionality will be realized by two dedicated components that will share interfaces with the WODM tool. The first one is the **SLA Manager (SLAM)** and it will be the component responsible for providing the SLAs, the KPIs and for establishing communication with the WODM tool, while the second one, the **Blockchain and Smart Contract (BC-SC)** platform, will be the component responsible for running the Blockchain on the various nodes, and also for deploying and executing the SCs and updating the KPIs. The WODM tool will request and receive predefined SLAs and KPIs from SLAM.

Once the SLA has been associated with KPIs, stakeholders and tasks and it is received by the SLAM from WODM, it stored and archived and then it is sent to the BCSC platform. There, it is translated into a SC and it is initiated on the appropriate nodes of the Blockchain. Then, WODM collects inputs from other components and updates the KPIs on the BCSC, the BCSC uses those values to calculate the SLA performance of the SCs. The SLA performance is calculated as the weighted average of the individual KPIs that are included in the specific SC. When the SC is updated, the result is sent back to WODM.

The SC functionality in COGITO project aims to **reduce the trust gap** between the involved stakeholders, and to communicate progress information securely and reliably as **a single source of truth**. Instead of trying to disruptively revolutionize the processes of a construction project, it will seamlessly be integrated into already existing mechanisms. More specifically, SCs will be combined with the very well-known concept of the Work Order, a document that outlines the details of a specific project. The innovation that will be introduced with the implementation of SCs, is the set of predefined KPIs that will be bundled with the Work Order to **monitor the progress** of each task. On the front end, and within the boundaries of human-machine interaction, parties responsible for the completion of a task, such as workers, projects managers, site managers, quality managers etc. will enter their inputs into the **Work Order Execution Assistance** (WOEA) application and the WODM tool. This light weighted tool aims to simplify and digitize data collection from the construction site. In a business-as-usual scenario of a construction progress, the party responsible for the management of the whole progress would also collect and store the data in a centralized repository. Instead, a decentralized approach will be followed in COGITO, using the DLT, where data will be jointly stored and shared on a distributed.

In this way, the COGITO project will be securing tight sealed integrity and data reliability without the necessity of a central authority, and it will be benefited in both technical and financial terms.

- From a **technical point of view**, this approach offers an outstanding advantage in terms of security. Empowered by cryptographic keys and a BFT engine it makes it almost impossible to maliciously interfere or hack any data stored on the chain. Additionally, it is impossible to randomly or by accident change or delete data, since blocks retain all the changes of state. By allowing a highly secure and tamper proof system of multiple nodes to store data, then the risk of failure to reach or interact with a single database due to a server malfunction or a network failure is also eliminated.

- From **a financial point of view**, Blockchain and SCs can significantly reduce the administrative cost of task completion verification and potentially, the transaction cost between the participating members. This effect is visible on a primary but also on a secondary level. The straightforward benefit can be derived from the automation mechanisms, the speed of transactions and the inherent reliability of the underlying technology, that will simplify and speed up timely procedures leading to an overall cost reduction. On a secondary level, by introducing a single source of truth mechanism, conflicts will be significantly reduced and delays that could otherwise result to a disastrous budget derailment can now be avoided.

2.4.2 KPIs and Smart Contract enabled SLAs

The innovation that is brought to the COGITO project by the Blockchain technology is the bundling of KPIs with stakeholders in a smart contract enabled SLA, that follows the rules and terms dictated by the work order. At the center of this approach is the SC, a piece of software that is programmed to automatically perform a set of actions based on predetermined and predefined KPI values. The KPIs should be carefully selected so that: a. they can provide a quantifiable and data driven indicator of task completion and b. they sufficiently cover all the aspects of a construction process. A collection of indicative KPIs is presented in Table 1. KPIs are provided by WODM and are calculated using inputs from WODM UI users.

The values are safely stored on the nodes of the BC-SC and based on them the overall performance of the SLA is calculated and provided back to WODM. For the overall performance of the SLA the weighted averaged is selected and the methodology is described in section 2.4.4.

Table 1 – Indicative KPIs and their description.

KPI	Description
Percentage of worked hours	Indicator that shows the progress of a task compared to the already planned schedule. Compares the actual worked hours on a task versus the scheduled ones. **Actual vs Planed**
Percentage of completed work	Completion of a specific task **Completed Milestones vs Remaining Milestones**
Percentage of labour/equipment downtime	Period in which a piece of equipment or a machine is not functional or cannot perform any work. It may be due to technical failure, machine adjustment, maintenance, or non-availability of inputs such as, labour, power. **Downtime hours vs total hours**
Percentage of available Workers	Indicator to measure if the contracted personnel is present on site. **Actual vs Planned**
Percentage of available equipment	Indicator to measure if the contracted equipment is present on site. **Actual vs Planned**
Percentage of passed site inspections	Indicator to be used as a point of reference regarding the completion of a task. **Number of inspections passed vs Total number of inspections**

Primarily, the KPIs should provide insights on the completion of a task, so that all the participating members are aware of the progress achieved so far. This can be done through the **Percentage of completed work**, a KPI that is defined as the ratio of the completed milestones versus the total number of planned milestones. The **Percentage of worked hours** could be used independently or complementary with the previous KPI. This indicator reveals how much of the planned time on a task have been consumed so far, and in order to be calculated the total amount of worked hours from all the workers should be summed. These two indicators could be used together in the same SLA. For example, in case a task has a Percentage of completed work at 50%, but the Percentage of working hours is at 90%, then it shows that the task is halfway completed but most of the planned hours have been consumed. This indicated that there were either some serious underperformance

issues or that the initial planning was poor. Many issues that could potentially affect the relationship between contractors and subcontractors are also taken into account. A common source of dispute could appear in case a subcontractor delivers fewer workers or equipment than what was initially contracted. Both the **Percentage of available Workers** and the **Percentage of available Equipment** indicators aim to resolve these types of conflicts. The **Percentage of labor/equipment downtime** is indicator that shows if available equipment or personnel is present but remains idle. This could be used in a plethora of situations to secure both contractors and subcontractors. For example, in case of poor weather conditions or other environmental factors, workers are present but not able to work. A subcontractor could also ask for a specific equipment downtime allowance to perform scheduled or unexpected maintenance. Finally, the **Percentage of passed site inspections** could be the indicators that when achieved at 100% then the completion of the task is confirmed.

2.4.3 Smart Contract Data Model

Once the SC is active on the blockchain, then it monitors the delivery of the service through the KPIs that are provided by WODM. Typically, the delivery of a service is issued and monitored through a hand-written document, the work order or the SLA, both are similar terms that describe a contractual agreement between two parties. They display all the information regarding the contracted parties and the specifications of the delivered task. However, for a work order/SLA to be translated into a SC and become an integrated element of the COGITO ecosystem, it should be converted from a conventional legal document into a machine-readable file/format. Therefore, the basic components of the work order/SLA should be identified and expressed accordingly, in a way that the enforcement of a legal agreement is only a matter of correct data entry and proper code execution. An overview of the information that describes the basic work order/SLA elements that will be translated into independent data entities for SC implementation is given below.

SLA Information: In order to be coherent from both human readers but also from a computer language perspective, the SLA entity should host some basic information. The first one is the unique identification number of the specific contract which is generated and delivered by the WODM tool, then the general description in text form to give an overview of the contents, the timestamp of the contract's creation time, and finally the starting date together with an ending date to present the validity period. The latter is of high importance due to the immutable and unalterable nature of the blockchain. After deployment any change would require changing the entire blockchain.

Task: The Task entity is the entity that links the SLA with the workflow task and the respective work order. Besides the identification number a text description is also included to provide more information to the human reader.

Party: An SLA is a contractual agreement that describes the delivery of the service, and the involved parties should be identified alongside their role, that can be a worker, a site manager, a project manager etc. The identity of the participating members will be cryptographically encrypted.

Service Level Objective: This is the key element of the SLA; the Service Level Objective (SLO) defines the level of the expected service delivery between the customer and the service provider. The SLO in order to be totally comprehensible should include the indicator to be monitored, the expected value of the indicator and the unit of measurement. A list of indicative KPIs is presented in Table 1 and inputs for their calculation will be received from the WODM tool. The person responsible for issuing the work order will also define the target value of the KPI on the SLA. For example, a project manager creates a work order for a specific task that involves three workers and should pass 3 inspections to be considered completed. The KPIs that could be used in this case are the *Percentage of passed site inspections*, the *Percentage of completed work* or even the *Percentage of available Workers*. The project manager should include as a participating member of the SLA (the Party entity above) herself, the three workers and the quality manager responsible for the inspections. Then those parties will use WOEA to report their progress and once the target values are achieved, meaning 3 out of 3 of passed inspections then the SLA will be considered completed.

Rules and Actions: This section is a logical sequence of the SLO section above as it is concerned with the KPIs and the KPI target value. It shows what rules need to be followed and what type of action needs to take place when the rule is satisfied. The attributes of this entity are two, the "rules" and the "actions". When the SC receives a

KPI input, then it compares the measurement with the target value ("rule") and if the condition is satisfied then an "action" takes place. The "rule" is simply one of the following options: "more than", "less than" or "equal to". In the example given above, let's assume that in order for the task to be completed then two things need to happen. First the workers need to report that their work is done and then the quality manager should inspect and give the green light. In SC language this translates to the following "rules", *Percentage of completed work* "equals to" 100% and *Percentage of passed site inspections* "equals to" 100%. In case the workers have all reported that the task is completed but the site has failed during one of the inspections, then the *Percentage of completed work* "equals to" 100% but the *Percentage of passed site inspections* is not "equal to" 100%, as it is 66% for 2 out of 3 passed inspection. Therefore, the action taken by the SC is to report that the task is not completed, and the work order is still pending. And it will remain in this status as long as one of the KPIs has not reached the contracted level.

Equipment: This field is optional and should be included only if the use of equipment is necessary for the completion of the specific task. The person responsible for issuing the SLA should also include the type of the equipment, the id of the equipment, alongside with the time period that the equipment should be available and on site. These fields could then be used to calculate KPIs such as *Percentage of available equipment* and *Percentage of equipment downtime*.

Based on the information presented above, Table 2 could be used as a template for the development of Smart Contract enabled SLAs for the COGITO project.

Table 2 – Data entities for Smart Contract enabled SLAs.

	Business Model Title
SLA Basic Information	SLA ID Star Date End Date Timestamp Description
Parties	Party ID Party Role
Service Level Objectives	SLO Indicator SLO Target value SLO Unit SLO Weighted Factor
Task	Task ID Task Description
Rules and Actions	Rule Action
Equipment	Equipment Name Equipment ID

The relationships between the aforementioned entities, alongside with information regarding their attributes, are depicted in Figure 4 as a conceptual data model. There is a composition relationship between the Party and the SLA entities, with two or more parties to be required to establish an SLA, at least one that has the role of the service provider, for example a worker, and at least one to be the service consumer, for example a project manager. One Task is assigned to each SLA. There is a one-to-many relationship between the SLO and the SLA entities to express one or more KPIs that are necessary to monitor whether the service is properly delivered or not. Similarly, there should be at least one "Rule-Action" pair for every SLO that is expressed. The Equipment entity is optional and is linked with the Task entity.

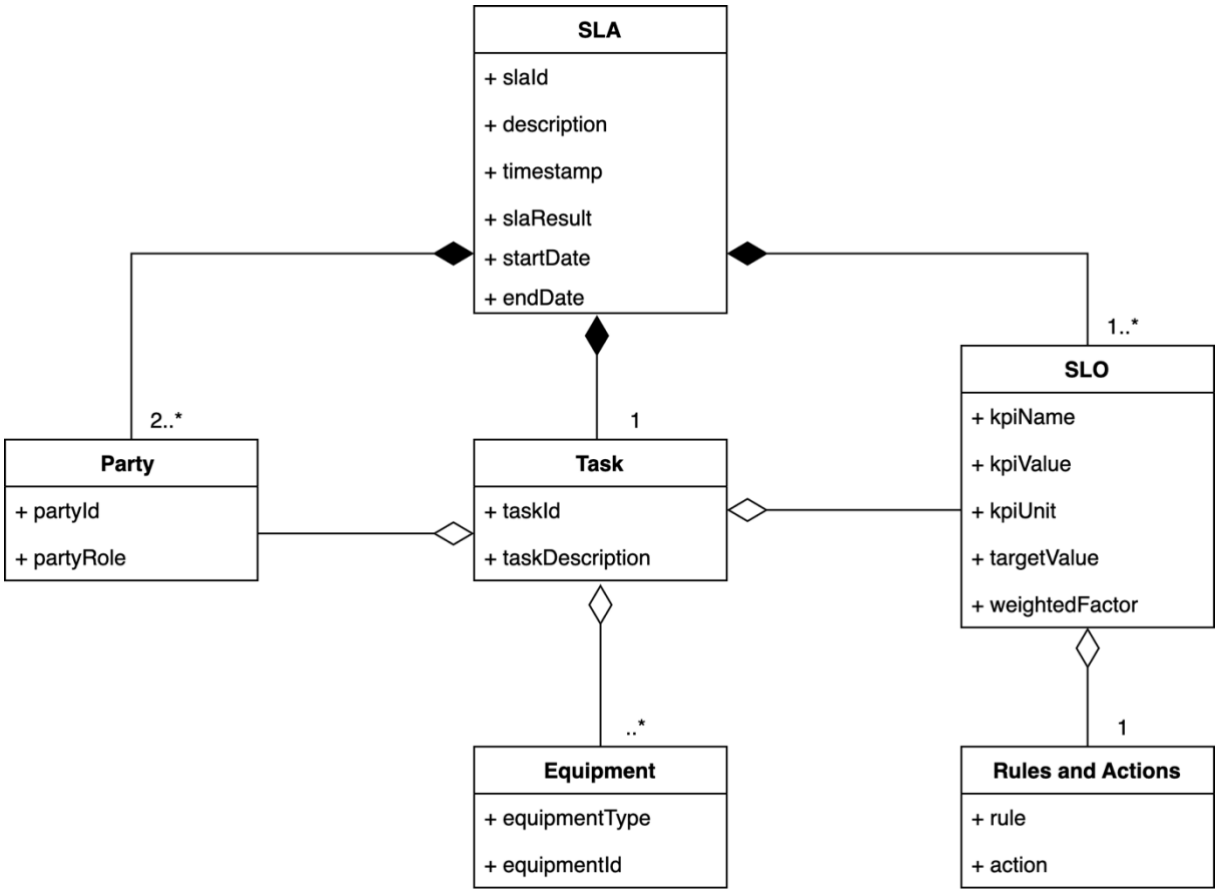


Figure 4 – SLA Conceptual Data Model

2.4.4 The Blockchain Platform

Once a SC has been deployed and initiated on the Blockchain then the users can start performing transactions. The SC in order to be called and start performing transactions has its own unique address. A “transaction” is an action that is initiated by a human user and refers to an exchange of data, and the "transaction request" is the term used to define the request for code execution. When the transaction is complete, meaning that the code of the SC has been executed according to the input data that were provided by the user, then there is a change of state on the Blockchain. Transaction requests on the Blockchain network, can be initiated by any authorized user through a node. To safeguard the Blockchain’s integrity and to prove that the transaction is not sent fraudulently, it needs to be signed with the private key of the user.

This process is also highlighted in Figure 5, the authorized user through the WODM UI or the WOEA tools reports the progress of the tasks (Figure 5.1). This action in Blockchain and SC terminology, is translated to a call to a unique address that is held by a SC. The data payload of the transaction consists of the updated KPI values that the user wants to report on the SC. This transaction then is broadcasted to the whole network waiting for validation. The validation process (Figure 5.2), among other things, will confirm whether the initiator is an authorized user, or if this transaction was initiated more than one times. After the consensus of the Blockchain nodes is achieved, the validation is considered completed, and the hashed transaction information is included in a “block” creating a tamper-proof record. As a final step (Figure 5.3), the Blockchain is updated by including the new block to the chain of blocks, and in this way the whole network has their own updated record of the digital ledger.

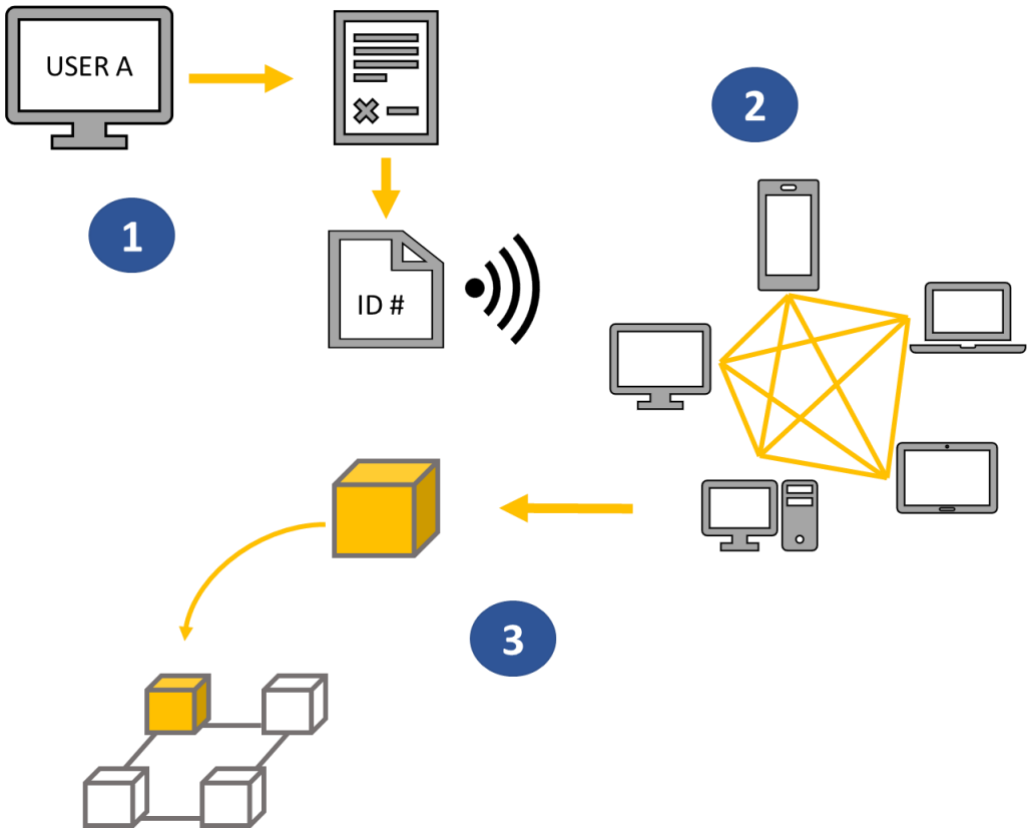


Figure 5 – Transaction on the Blockchain using a Smart Contract.

The copies of the Blockchain on every node of the entire network are identical, and they are updated simultaneously to ensure the integrity of the information that is held, and the equal and fair access of the participants. This means that in any given moment every user has her own, but the exactly same, copy of Blockchain and therefore, everyone has access to every previous interaction and transaction between the rest of the authorized members. The record of transactions on the COGITO Blockchain platform will have a specific structure as it can be seen on indicative example in Table 3. For every individual transaction, the identity of the initiator will be referred, along with the Contract ID and the KPI that the initiator updated and the result of the SC upon execution of the transaction.

Table 3 – Indicative example of transaction records on Blockchain.

Initiator ID	Contract **ID**	Updated KPI	SLA Performance [%]
003429b07f4a4f54be37	2f8b6c4006ec67fe93a6	KPI1=0.33	50
00badc1766024a8fac11	01411e0ca3aa477c8c6b	KPI2=0.12	70
692fe9ad06fab9ac72b8	29a48eca099eafa693b7	KPI3=0.70	30
7462c8d9be23ad63b0cc	7d8cbe89bacd5038bca2	KPI4=0.68	100

The SLA Performance is indicator that shows the overall performance of the specific SLA. It is calculated as the weighted average of the KPIs bundled to the SLA as follows:

$$SLA\ Performance = \frac{\sum_{i=1}^{n} w_i KPI_i}{\sum_{i=1}^{n} w_i} \quad .$$

The weighted average was selected as it can provide more flexibility and represents in a more accurate way the importance of each KPI in the completion of each task. For example, the **Percentage of available workers** is useful to show whether all the contracted workers were present during construction, and solve potential

disputes with subcontractors, but the **Percentage of Completed Work** better represents the actual progress of the task. For simplicity, in the current version of the deliverable, we assume that all weights are equal to 1.

3 Blockchain and Smart Contract Platform

3.1 Prototype Overview

The Blockchain-Smart Contracts Platform demo has its functionality features based on D2.1 (Stakeholder requirements for the COGITO system) and its structural design based on D2.4 (COGITO system architecture v1). The conceptual UML component diagrams of the two components, the SLAM and the BC-SC are presented in Figure 6 and Figure 8 respectively, alongside with their dedicated interfaces and the tools that were used for their development. The SLAM is a component that provides a database with already designed SLA templates and creates bundles of SLAs, KPIs, rules and stakeholders that will be translated to SC. It consists of three subcomponents:

- **Smart Contract Manager**: It is the component responsible for establishing communication with WODM and to provide the SLA templates with the predefined KPIs. The authorized WODM user will bundle the stakeholders responsible for undertaking or supervising a task, alongside with the KPIs that are suitable for monitoring. It will also store locally the created SLA bundles.
- **SC Orchestrator**: This component acts as the pathfinder for the population of the nodes of the Blockchain. It is responsible for identifying the appropriate Blockchain Network to initiate the deployment of the SC.
- **Security** subcomponent that provides the means for user authentication and authorization, so that only the authorized users can have access to SLA templates and bundle SLAs with KPIs and stakeholders. It is based on Network IP Security that acts as a firewall allowing only specific IPs to access the specific tool.

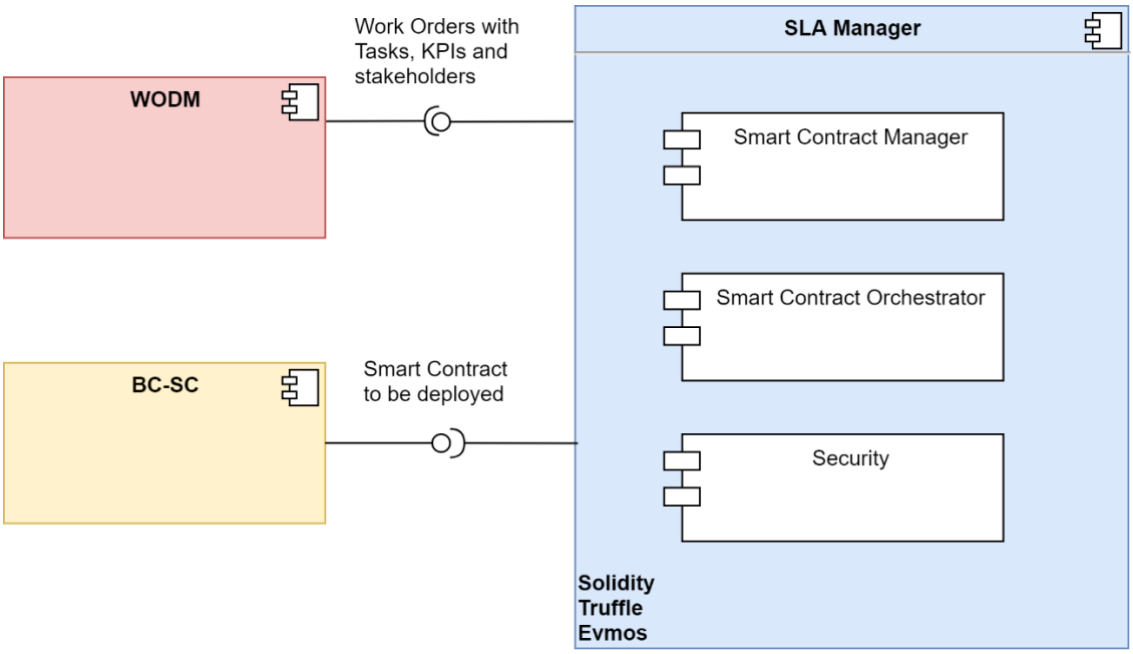


Figure 6 – SLA Manager UML component diagram.

For the development of SLA Manager, Solidity alongside with Truffle and the Evmos frameworks have been used. These tools will be thoroughly analysed in a following section. Moreover, the SLA Manager provides a number of API Services to the WODM tool. Through these services the WODM user can retrieve the available KPIs, submit the newly created SLA bundles and retrieve archived SLAs from the repository of the SLAM. The complete SLA bundles are sent to the BC-SC component to be initiated as SC. For a more detailed overview of the interaction between the components but also for the role of the SLAM subcomponents during the workflow planning phase, a sequence diagram is presented in Figure 7. Specifically Figure 7 describes the interactions number 27, 28, 32, 33 and 34 of UC1.1 “Efficient and Detailed project workflow planning using the project's construction schedule

and as-planned BIM model" that is presented in D2.5 chapter 3.1 together with the interaction between the SLAM subcomponents.

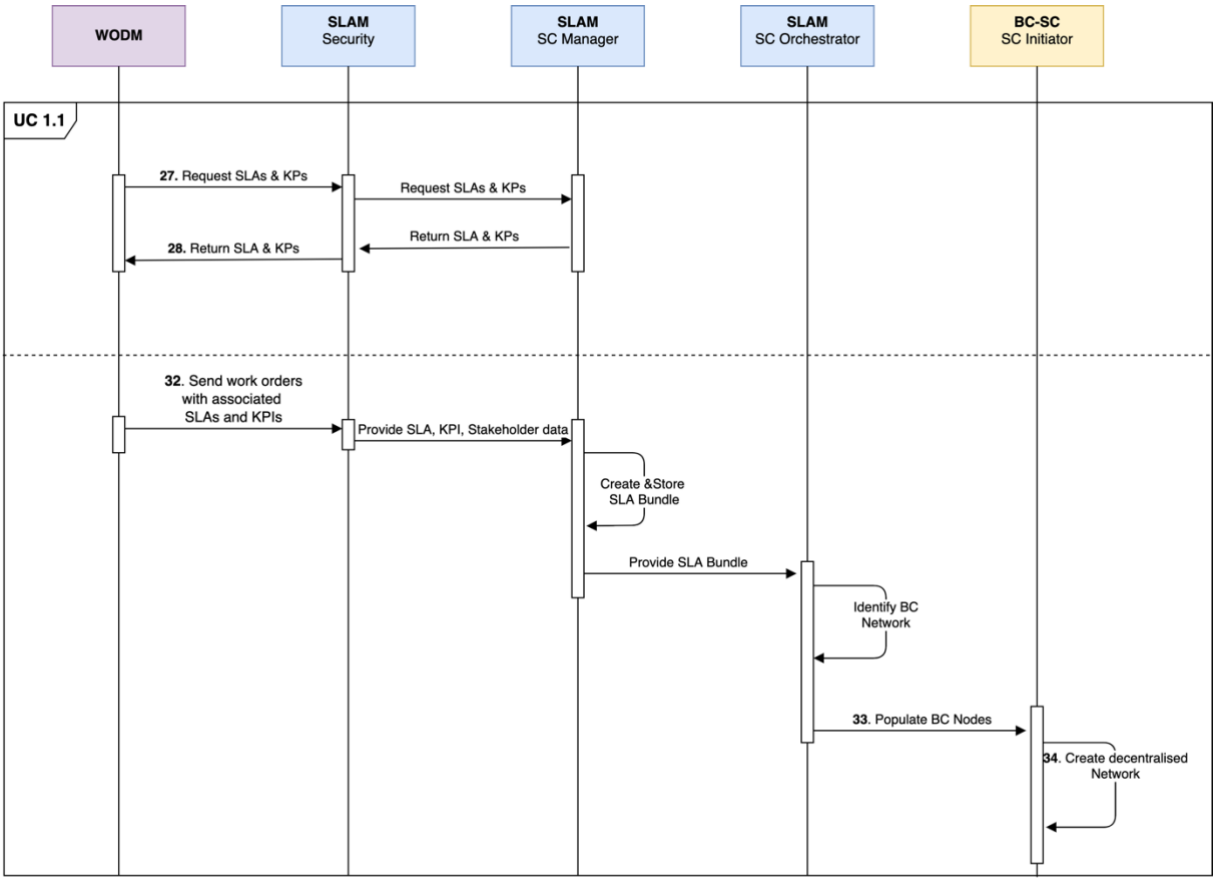


Figure 7 – UML Sequence diagram presenting SLA creation and interactions between WODM, BC-SC and SLAM subcomponents, UC1.1 numbering have been used.

During interaction 27 "Request SLAs and KPIs", the authorized WODM user requests and receives (interaction 28) the available KPIs from the SC Manager subcomponent, which keeps this information stored on a local repository. The following step is interaction 32 in which WODM sends to the SLAM the SLAs bundled with the KPIs and the stakeholders. These bundles are then sent to the SC Manager subcomponent, which creates, timestamps and stores locally the complete SLA. Then it is forwarded to the SC Orchestrator, the component responsible for the population of the Blockchain nodes with the new SLA. Upon reception of the SLA from each node (interaction 33), the BC-SC Initiator translates the SLA into SC and initiates it on the Blockchain Network (interaction 34).

The second component, the BC-SC Platform, is an immutable database system that will initiate and execute the SCs created and received from the SLAM. It will also store all the KPI values that are necessary for the operation of the SCs, alongside with each SLA Performance value. In this way a complete and trustworthy record of the stakeholders' inputs, the KPI values and the SLA Performance is kept. The BC-SC component and the three associated subcomponents are presented in Figure 8 below. For the development of the BC-SC platform the Cosmos/Tendermint ecosystem was used. These frameworks were selected as they are highly scalable but also modular, so they allow for increased usability and interoperability compared to previous technologies. The Cosmos/Tendermint ecosystem relies on the PoS consensus mechanism for the validation of transactions. Compared to the PoW that requires large amount of energy, PoS is considered to be more environmentally friendly and more cost-efficient. The software tools for the development of the demo are described in section 3.2.

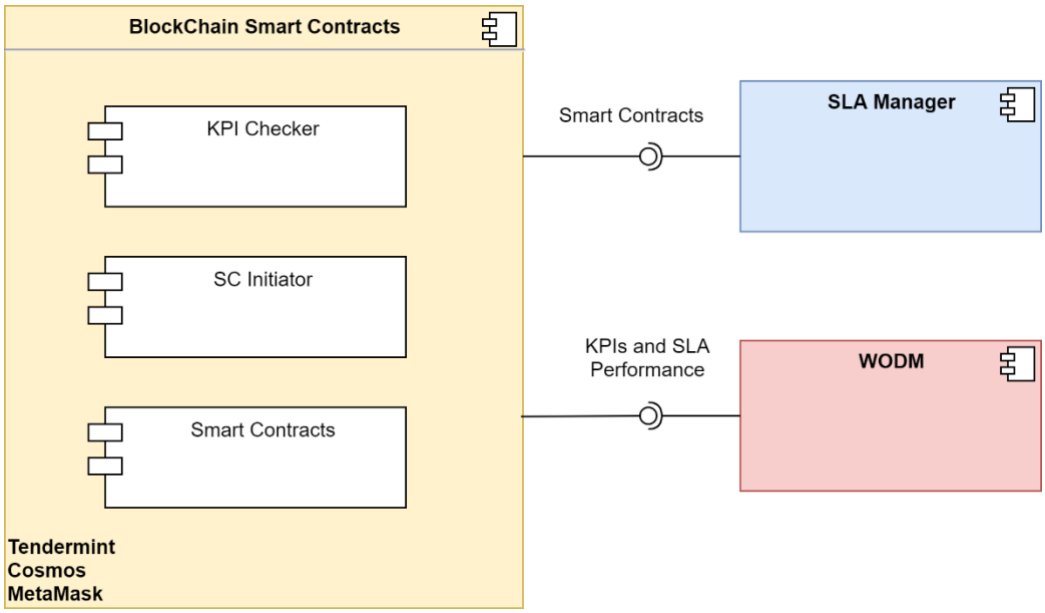


Figure 8 – UML component diagram of BC-SC Platform.

The three subcomponents of the BC-SC Platform are:

- **Smart Contract Initiator:** This is the subcomponent that receives the complete SLAs from the SLAM, and initiates the SC instance on every node of the Network
- **KPI Checker**: This is the subcomponent that communicates with WODM in order to receive updated KPIs, once the KPIs are validated, they are send to the Smart Contracts subcomponent.
- **Smart Contracts:** This is the Blockchain component that is used for the deployment and execution of SCs. Once a validated KPI is received from the KPI Checker, the SC runs in every node and produces the SLA Performance. The KPI value, together with the authorized user and the current SLA Performance value are stored in the Blockchain. The WODM can access the Smart Contracts subcomponent and retrieve these values.

A detailed overview of the interactions that take place during the update of KPIs as it is described in D2.5 (chapter 3.2) between the WODM and the BC-SC Platform are presented below in Figure 9. The WODM provides the updated KPIs to the KPI Checker to be validated and forwarded to the Smart Contract subcomponent. There they are stored, and the SLA Performance is calculated. The WODM user then receives the SLA Performance through a dedicated API service that is described in section 3.3.

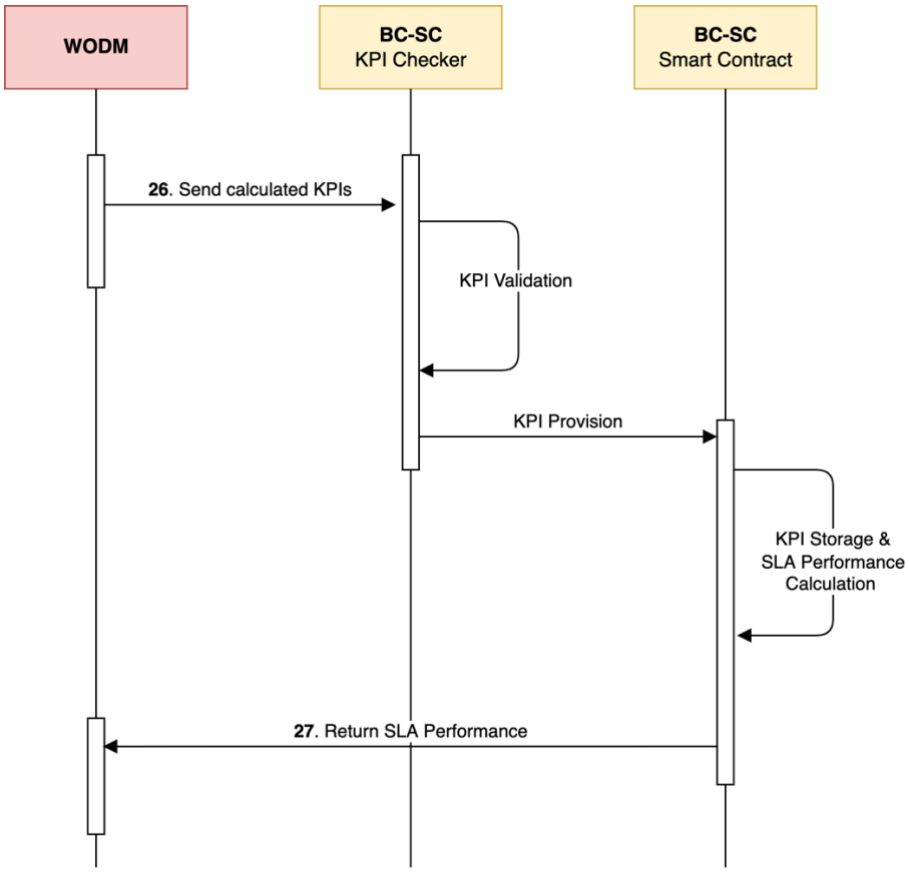


Figure 9 – UML Sequence Diagram presenting WODM – BC-SC Platform interactions for KPI update and SLA Performance provision, UC1.2 numbering have been used.

3.2 Technology Stack and Implementation Tools

3.2.1 Tendermint

At the foundation of this prototype lies the Tendermint[1] BFT state machine replication software. BFT is the software that allows a decentralized system to continue its operation even if some nodes fail to respond or act maliciously. It is a consensus protocol that is primarily used for data verification, and it has the ability to operate efficiently and effectively even if 1/3 of the total nodes are not responding due to technical difficulties or act maliciously. In its essence a state machine is an abstract machine that when it receives an input, then alters its state and moves to a new one. This state of change is called a transition, and can be the result of a SC execution. So, by receiving a given input, a state machine performs a transition and produces new outputs. Therefore, a Blockchain is not just the database itself, but all of the consensus algorithms that allows for secure communication of the new state among the nodes. The Tendermint as a software package and the underlying technology of Cosmos SDK is responsible for managing three very important aspects. Primarily, it is responsible for the propagation of the new state among the nodes of the peer-to-peer network. On a second level, it guarantees that all the nodes achieve consensus, meaning that they agree and store the specific state. And finally, it has the application interface to allow the nodes to update their state based on the performed transactions. The first two are part of the Tendermint Core and the last one the is the Application Blockchain Interface (ABCI). One of the great advantages offered by Tendermint is its architecture. The application layer responsible for the development of specialized software components can be abstracted from the core components, the Consensus

[1] https://tendermint.com

and the Networking. This allows the easier development of custom application software to handle the transactions as it can be seen on Figure 10.

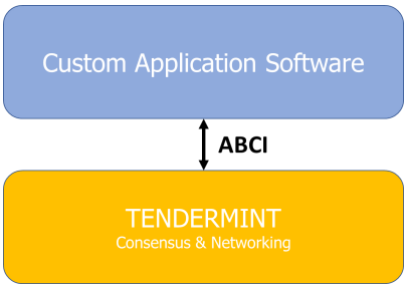


Figure 10 – Conceptual representation of Tendermint operational architecture.

3.2.2 Cosmos SDK

The second tool that was used for the development of the demo was the Cosmos SDK[2]. It is a framework that operates on top of the Tendermint core functionalities, meaning the consensus and the networking, and it is used for building custom application Blockchains. From those underlying functionalities it also inherits the PoS consensus mechanism and, it is ideal for fast and reliable development of Blockchain applications due to the large variety of modules that are available. This feature makes it widely popular across the industry and it is considered to be one of the most rapidly growing ecosystems. Since the core functionalities are handled by default from the underlying Tendermint framework, developers are free to focus their efforts solely on the business logic of their application. Using Cosmos SDK, they can put together existing SDK modules or create their own custom ones, to build a Blockchain that serves their own use case. Every Blockchain needs to perform specific actions which are common for most applications, such as creating and managing accounts, staking, token creation and management. For these actions the Cosmos SDK provides a wide range of existing modules that can be tuned if necessary. In case that someone wants to add an extra functionality or an innovative feature then the development of a new module, that will be interoperable with the existing ones, can be done using Go programming language. Another feature that is inherited by Tendermint engine and can be utilized on the application layer through Cosmos SDK, is the Inter Blockchain Communication protocol (IBC), which is the standard for Blockchain interoperability. Through IBC different Blockchains are able to interconnect and communicate with each other and exchange data or tokens. This turns a previously isolated system, the Blockchain, and makes it a part of an interconnected network, widening the scope of potential applications. As it can be seen in Figure 11, Cosmos Hub is a Blockchain that was designed and built specifically for this purpose, to allow other blockchains to interoperate. Tendermint, as the underlying framework (red color), is common in every Blockchain, and despite that each one serves a different application (different colors on top of Tendermint) they are all built using Cosmos SDK, and are interconnected with IBC through Cosmos Hub.

[2] https://v1.cosmos.network/sdk

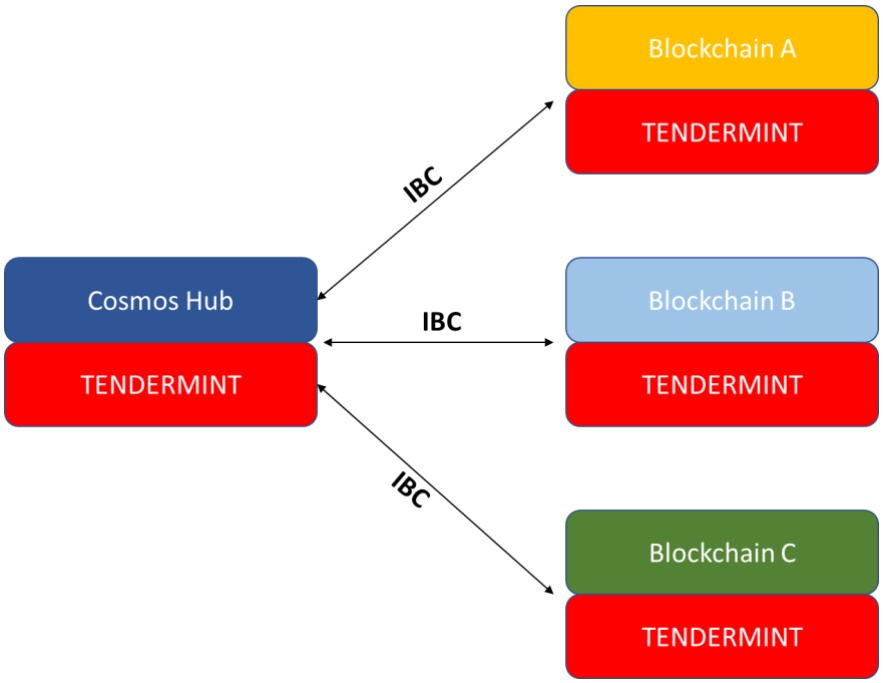


Figure 11 – Conceptual representation of Inter Blockchain communication.

3.2.3 Evmos

Evmos[3] is the Ethereum Virtual Machine (EVM) hub of Cosmos Network. In general, EVM is a software platform that is used for developing decentralized applications like SCs on the Ethereum. Evmos is a framework that is built using Cosmos SDK, and as mentioned in section 3.2.2 of the present document, it runs on top of the Tendermint consensus engine. As a result, it has the ability to offer the well-tuned and widely applied SC engine of the Ethereum, while at the same time it takes advantage of Tendermint's PoS implementation for security and fast finality. Within the concept of the COGITO Project and the current demonstrator, the Evmos framework will be used together with other existing and custom-made modules of Cosmos SDK to offer a reliable and scalable SC functionality.

3.2.4 Solidity

Solidity[4] is a programming language that belongs to the object-oriented genre, and its main function is to write SCs on various blockchain platforms. It is tightly associated with the Ethereum since their core designers have been involved with the development of the platform. Solidity was designed to help developers create SCs on the EVM, and in the Cosmos environment this is achieved through the Evmos. From a technical point of view, it has been influenced by other scripting languages like JavaScript, and it also shares some similarities with C++ and Python.

3.2.5 Truffle

Truffle[5] is a very flexible development environment and testing framework for the creation of SC on the EVM pipeline. It offers built-in smart contract compilation capabilities alongside with automated contract testing for quick and easy development.

[3] https://evmos.dev/
[4] https://docs.soliditylang.org
[5] https://trufflesuite.com/

3.3 API Documentation

This section describes a preliminary version of the interfaces that have been briefly described in D2.4, Chapter 4. Tables 4-6 describe the services between the WODM and SLAM, SLAM and BC-SC platform and BC-SC platform and WODM respectively.

Table 4 – getSLA API Service SLAM - WODM.

Service Name	getKPIList
Description	Request by user to retrieve available KPI
Information sender	SLAM
Information receiver	WODM
Communication protocol	HTTPS
Method type	GET
Active URL	<baseURL>/ getKPIList
Request Parameters/Request Body	-
Response	{ "KPI": ["Percentage of worked hours", "Percentage of completed work" , "Percentage of labor downtime" , "Percentage of equipment downtime" , "Percentage of available Workers" , "Percentage of available equipment" , "Percentage of received materials" , "Percentage of used materials" , "Percentage of passed site inspections"], }

Table 5 – initiateSLA API Service WODM - SLAM.

Service Name	initiateSLA
Description	WODM user submits the SLA with the associated KPIs, stakeholders and equipment (if available)
Information sender	WODM
Information receiver	SLAM
Communication protocol	HTTPS
Method type	POST
Active URL	<baseURL>/ initiateSLA
Request Parameters/Request Body	{ "slaID":"SLA ID", "parties": [{ "partyId": "default partyId", "partyRole": "default partyRole" }], "sla_startDate": "default start date", "sla_endDate": "default end date", "sla_description": "default description", "task": [{ "taskId": "default task ID", "equipmentID":"equipment ID", "partyId": "default partyId" }], "serviceLevelObjectives": [{ "KPI":"selected kpi", "target_value":"selected target value", "rule": "more than / less than / equal to",

	"KPI_value":"current value",
	"taskId": "default task ID",
	"weighted factor":1
	}],
	"equipment":[{
	"equipmentID":"equipment ID",
	"name": "resource name",
	}]
	}

Response	The respond is a JSON with available SLAs and KPIs.

Table 6 – getSLAbyUser API Service SLAM - WODM.

Service Name	getSLAbyUser
Description	Request to retrieve SLA by user, that have been created within a specific timeframe (optional)
Information Sender	SLAM
Information Receiver	WODM
Communication Protocol	HTTPS
Method type	GET
Active URL	<baseURL>/getSla?partyId=default
Request Parameters/Request Body	partyId
Response	The respond is a JSON with available SLAσ.

Table 7 – iniateSC API Service SLAM – BCSC.

Service Name	iniateSC
Description	Sends the SLA, KPI, stakeholder bundle to the BC-SC to be initiated as a smart contract.
Information Sender	SLAM
Information Receiver	BC-SC
Communication Protocol	HTTPS
Method type	POST
Active URL	<baseURL>/ iniateSC

Request Parameters/Request Body	{
	"slaID":"SLA ID",
	"timestamp":"default timestamp",
	"parties": [{
	"partyId": "default partyId",
	"partyRole": "default partyRole"
	}],
	"sla_startDate": "default start date",
	"sla_endDate": "default end date",
	"task": [{
	"taskId": "default task ID",
	"equipmentID":"equipment ID",
	"partyId": "default partyId"
	}],
	"serviceLevelObjectives": [{
	"KPI":"selected kpi",
	"target_value":"selected target value",
	"rule": "more than / less than / equal to",
	"KPI_value":"current value",

	"taskId": "default task ID",
	"weighted factor":1}],
	"equipment":[{
	"equipmentID":"equipment ID",
	"name": "resource name"
	}]
	}

Response	The response of the request is {"status":"failed"} on failure or {"status":"success"} on success

Table 8 – updateKPIbyContract API Service WODM – BCSC.

Service Name	**updateKPIbyContract**
Description	Update the KPIs of a given SLA
Information Sender	WODM
Information Receiver	BC-SC
Communication Protocol	HTTPS
Method Type	POST
Active URL	<baseURL>/ updateKPIbyContract

Request Parameters/Request Body	Content-type: application/json
	{
	"partyId": "default partyId",
	"slaID":"SLA ID",
	"timestamp":"default timestamp",
	"updatedKPI":[{
	"KPI":"selected kpi",
	"KPI_value":"current value",
	"taskId": "default task ID"
	}]
	}

Response	The response of the request is {"status":"failed"} on failure or {"status":"success"} on success

Table 9 – getSlaResult API Service BCSC – WODM.

Service Name	**getSlaResult**
Description	Retrieve the result of a specific SLA
Information Sender	BC-SC
Information Receiver	WODM
Communication Protocol	HTTPS
Method Type	GET
Active URL	<baseURL>/ getSlaResult? slaId= defaultslaId
Request Parameters/Request Body	slaId

Response	Content-type: application/json
	{
	"slaID":"SLA ID",
	"taskId": "default task ID",
	"timestamp": "default timestamp",
	"slaPerformance": "default performance",
	}

Table 10 – getKPIResult API Service BCSC – WODM.

Service Name	getKPIResult
Description	Retrieve the value KPI of a specific SLA
Information Sender	BC-SC
Information Receiver	WODM
Communication Protocol	HTTPS
Method Type	GET
Active URL	<baseURL>/ getKPIResult? slaId = defaultslaId& taskId= default&sloKpiName=default
Request Parameters/Request Body	slaId taskId sloKpiName

Response	Content-type: application/json
	{
	"slaID":"SLA ID",
	"taskId": "default task ID",
	"timestamp": "default timestamp",
	"KPI_value": "current value"
	}

3.4 Licensing

Apache License 2.0 – This is a permissive license whose main conditions require preservation of copyright and license notices. Contributors provide an express grant of patent rights. Licensed works, modifications, and larger works may be distributed under different terms and without source code.

3.5 Installation Instructions

The SLAM and BC-SC components are offered to the COGITO applications as web services; as such, no file download, installation, maintenance or other related operation is to be performed by entities other than its creators. For the sake of completeness, guidelines on how to setup the environment for both components are provided in Annex 1.

3.6 Usage Walkthrough

The BC-SC platform aims to reduce the trust gap between the involved stakeholders by introducing an innovative reporting system on the already well-established concept of work order execution. As it is theoretically introduced in section 2.4.2 the progress of the work order execution is monitored and reported based on a series of KPIs (Table 1). Both components of the Blockchain Platform, the SLAM and the BC-SC, do not have a dedicated UI and they are operated through the WODM UI using a series of REST APIs, as it is described in section 3.3. Since the WODM UI is not yet developed and functional, this section will demonstrate the operation of SLAM and BC-SC components using the Postman[6] software.

The SCs are bundles of users, equipment, tasks and KPIs. The first three is information already known to the WODM user, so to create a SC the WODM user requests the available KPIs from the SLAM component using the GET command on the **getKPIList** as it is seen in Figure 12.

[6] https://www.postman.com

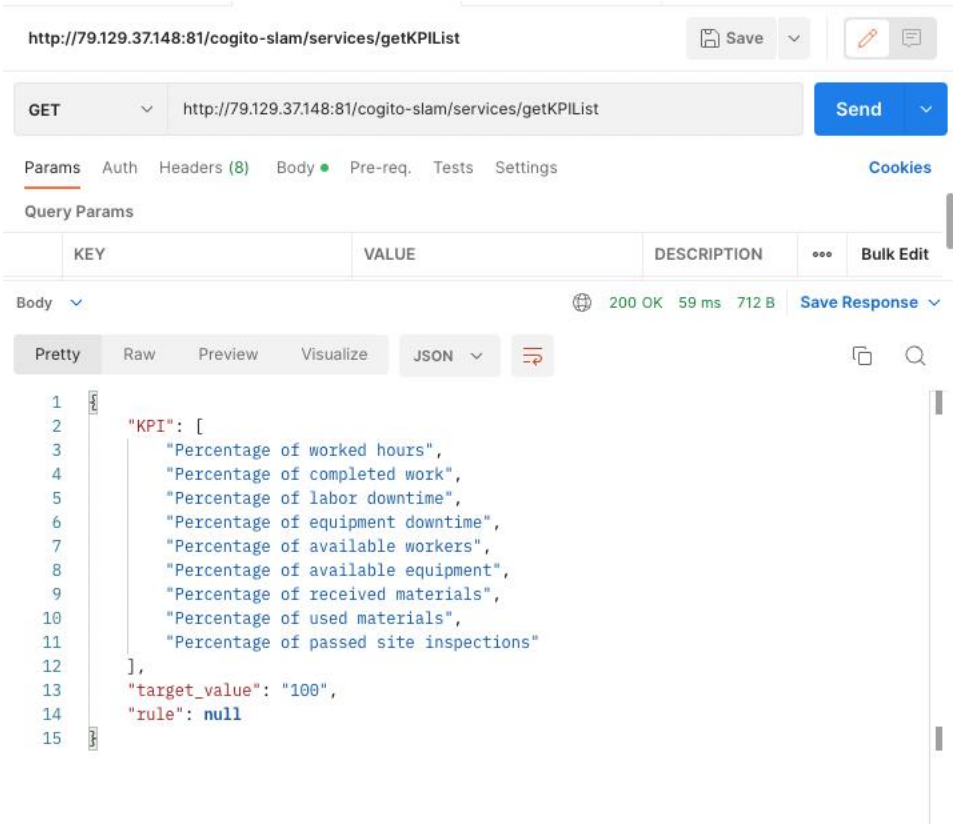


Figure 12 - Postman sample of the getKPIList service

Upon retrieval of the available KPIs for the specific project the WODM user can introduce the rest of the elements that are necessary for the creation of the SC. For our Usage Walkthrough example data from the school project have been used. Since there was not an already prepared example from the WODM tool a JSON file was prepared manually according to the data model that has been introduced during the UC1.1 Working Group. An overview of the JSON file can be seen in Figure 13. This JSON file is a work order digitized in a format readable from the SLAM.

```
{
    "slaID":"5dfe0io-5tb0-4fb1-ac3d-e1b164ce1dca",
    "timestamp":"2022-11-05T09:30:00",
    "parties":  [
        { "partyId": "7", "partyRole": "Finisher"},
        { "partyId": "8", "partyRole": "Worker"},
        { "partyId": "9", "partyRole": "Welder"},
        { "partyId": "10", "partyRole": "Surveyor"},
        { "partyId": "12", "partyRole": "Foreman" },
        { "partyId": "13", "partyRole": "Crane operator" }],
    "sla_startDate": "2022-11-01T08:00:00",
    "sla_endDate": "2022-11-01T17:00:00",
    "tasks": [
        { "taskId": "23gqc", "equipmentID": [], "partyId": ["12"]},
        { "taskId": "46gqc", "equipmentID": [], "partyId": ["12"]},
        { "taskId": "16vqc", "equipmentID": [], "partyId": ["12"]},
        { "taskId": "45vqc", "equipmentID": [], "partyId": ["12"]},
        { "taskId": "4gqc", "equipmentID": [], "partyId": ["12"] },
        { "taskId": "17gqc", "equipmentID": [], "partyId": ["12"]},
        { "taskId": "11", "equipmentID": ["6"], "partyId": ["7", "9"]},
        { "taskId": "12", "equipmentID": ["1", "5"], "partyId": ["8", "13"]},
        { "taskId": "11vqc", "equipmentID": [], "partyId": ["12"]},
        { "taskId": "13", "equipmentID": [], "partyId": ["12"]},
        { "taskId": "15", "equipmentID": ["6"], "partyId": ["7", "9"]},
        { "taskId": "16", "equipmentID": ["1", "5"], "partyId": ["8", "13"]},
        { "taskId": "17", "equipmentID": [], "partyId": ["8", "10", "13"]},
        { "taskId": "36vqc", "equipmentID": ["2","6"], "partyId": ["12"]},
        { "taskId": "26gqc", "equipmentID": ["4"], "partyId": ["12"]},
        { "taskId": "12gqc", "equipmentID": ["3","4"], "partyId": ["12"]},
        { "taskId": "9vqc", "equipmentID": [], "partyId": ["12"]} ],
    "serviceLevelObjectives": [
        {"KPI":"Percentage of completed work",
        "target_value":"100",
        "rule": "equal to",
        "KPI_value":"0",
        "taskId": ["9vqc","12gqc","26gqc","36vqc","11","12","13","15","16","17", "23gqc","46gqc","45vqc","4gqc","17gqc","11vqc"],
        "weighted factor":1}],
    "equipment":[
        { "equipmentID": "1", "name": "Truck mounted concrete boom pum" },
        { "equipmentID": "2", "name": "Concrete mixer truck"},
        { "equipmentID": "3", "name": "Dump truck"},
        { "equipmentID": "4", "name": "Excavator"},
        { "equipmentID": "5", "name": "Truck mounted crane"},
        { "equipmentID": "6","name": "Concrete vibrator machine"}]
     }
```


Figure 13 - JSON file of school project SLA.

This SLA involves 5 participating members, each one with a unique ID and separate role in the construction process, namely finisher, worker, welder, surveyor, foreman and crane operator (lines 5-10). There are also 6 pieces of equipment: a truck mounted concrete boom pump, a concrete mixer truck, a dump truck, an excavator, a truck mounted crane, and a concrete vibrator machine (lines 39-44). The SLA also includes 17 tasks (lines 14-30), for each task one or more parties have been registered, and depending on the nature of the task some pieces of equipment have been included. For this example, the evaluation of the SLA progress will be monitored through the "Percentage of completed work" KPI (line 32), that was selected from the available KPIs presented on Table 1. For each task one or more KPIs can be selected and assigned. In this case, to consider the task completed the value has to be equal to 100% (lines 33 and 34), while the current value is 0 (line 35) since the SLA has not started yet. For simplicity reasons, all tasks will be monitored through the same KPI (line 36) and are all considered to be equally important for the completion of the task since they have the same weighted factor (line 37).

This demonstrates how all the important information of a work order have been organized to become SC ready, in the exact format that the SLAM will be receiving information from the WODM tool. When the above JSON is used with the POST command on the **initiateSLA** service (Figure 14), then the WODM user will receive a success message with a small description and the address of the contract one the BC-SC. This indicates that the SLA has been received and it is a registered SC on the Blockchain platform. The Swagger of this service is presented on Figure 15.

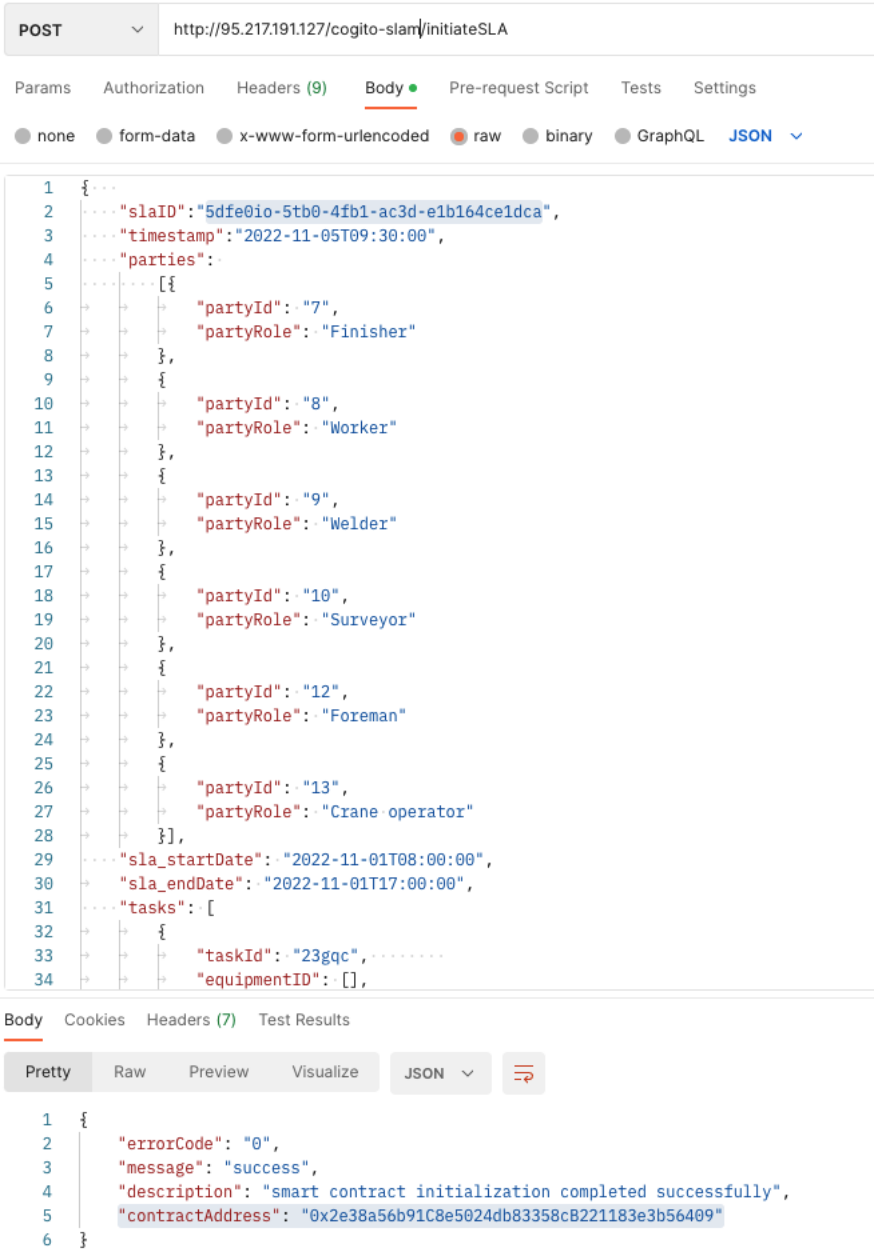


Figure 14 - Example of the initiateSLA service using Postman

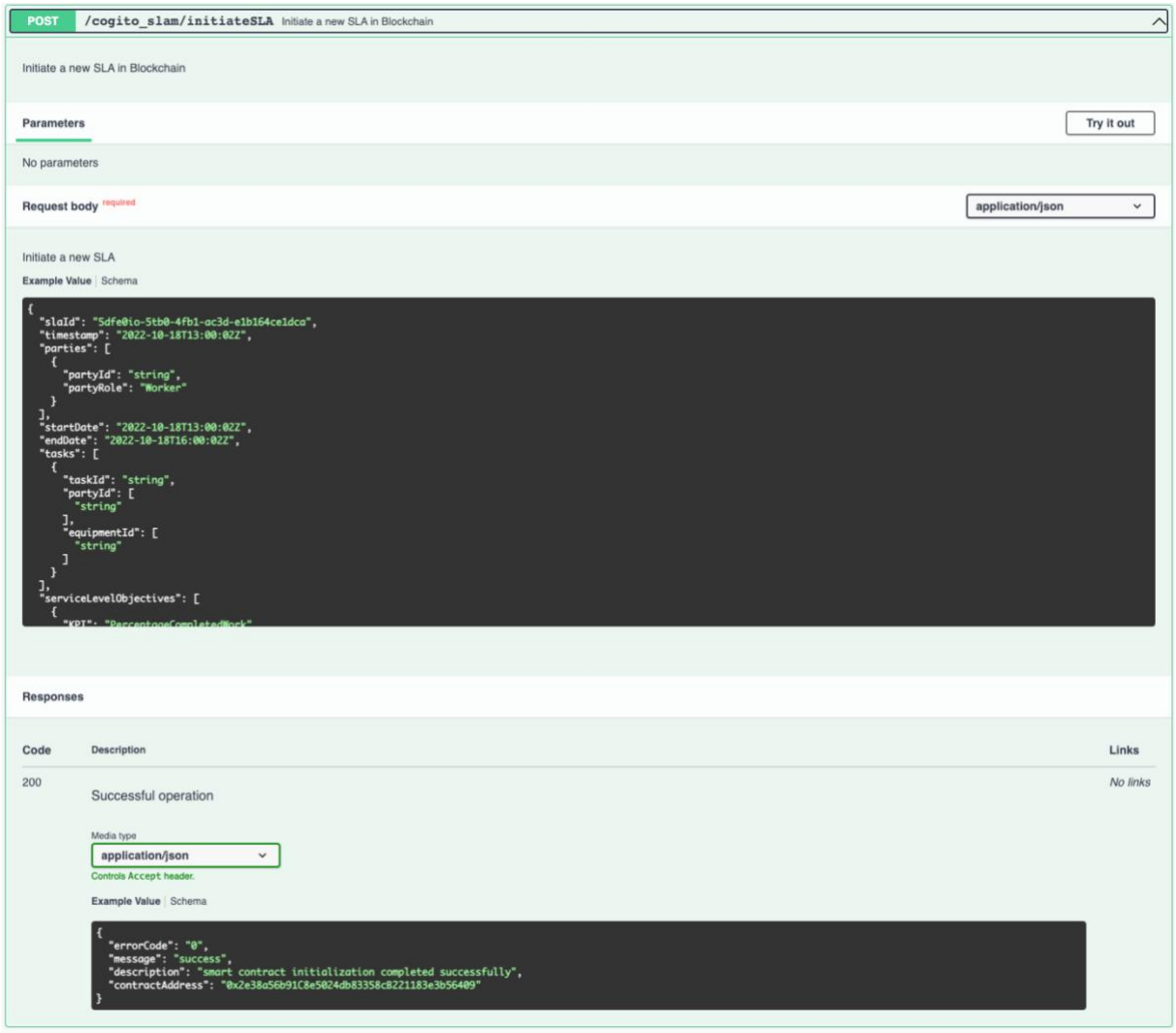


Figure 15 - The Swagger of initiateSLA.

Then the WODM user can start interacting with the SC using the WODM UI. For example, by using the GET command with the **getSLAbyUser** service and providing the party ID, the WODM user can retrieve a list with all the contracts that include the specific party as it can be seen in Figure 16 and the Swagger on Figure 17. In the current example as it is presented, the user with the party ID "7" retrieves the full list of information regarding the SCs that have been registered under this ID (currently only one).

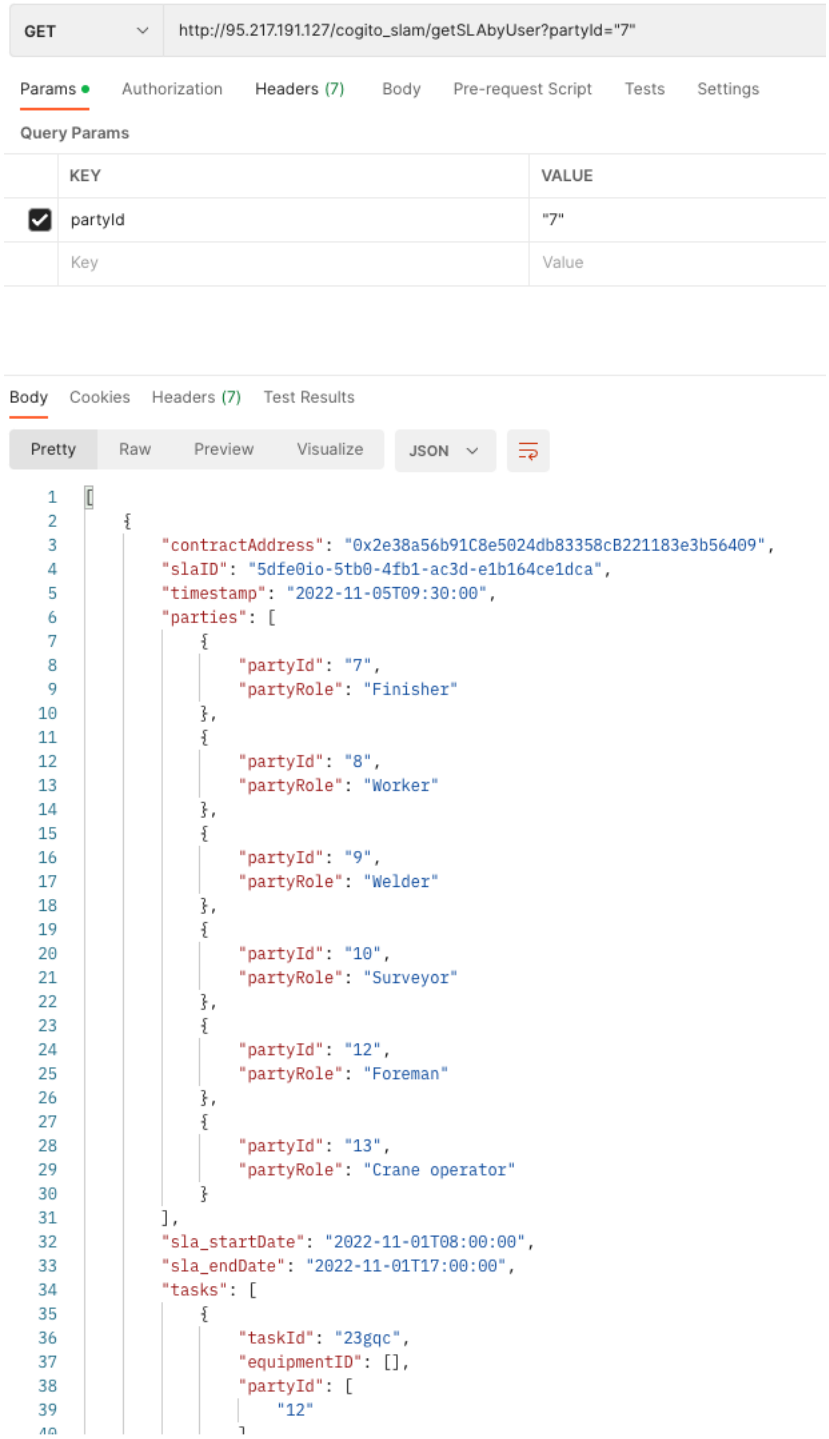


Figure 16 - Example of the getSLAbyUser service using Postman

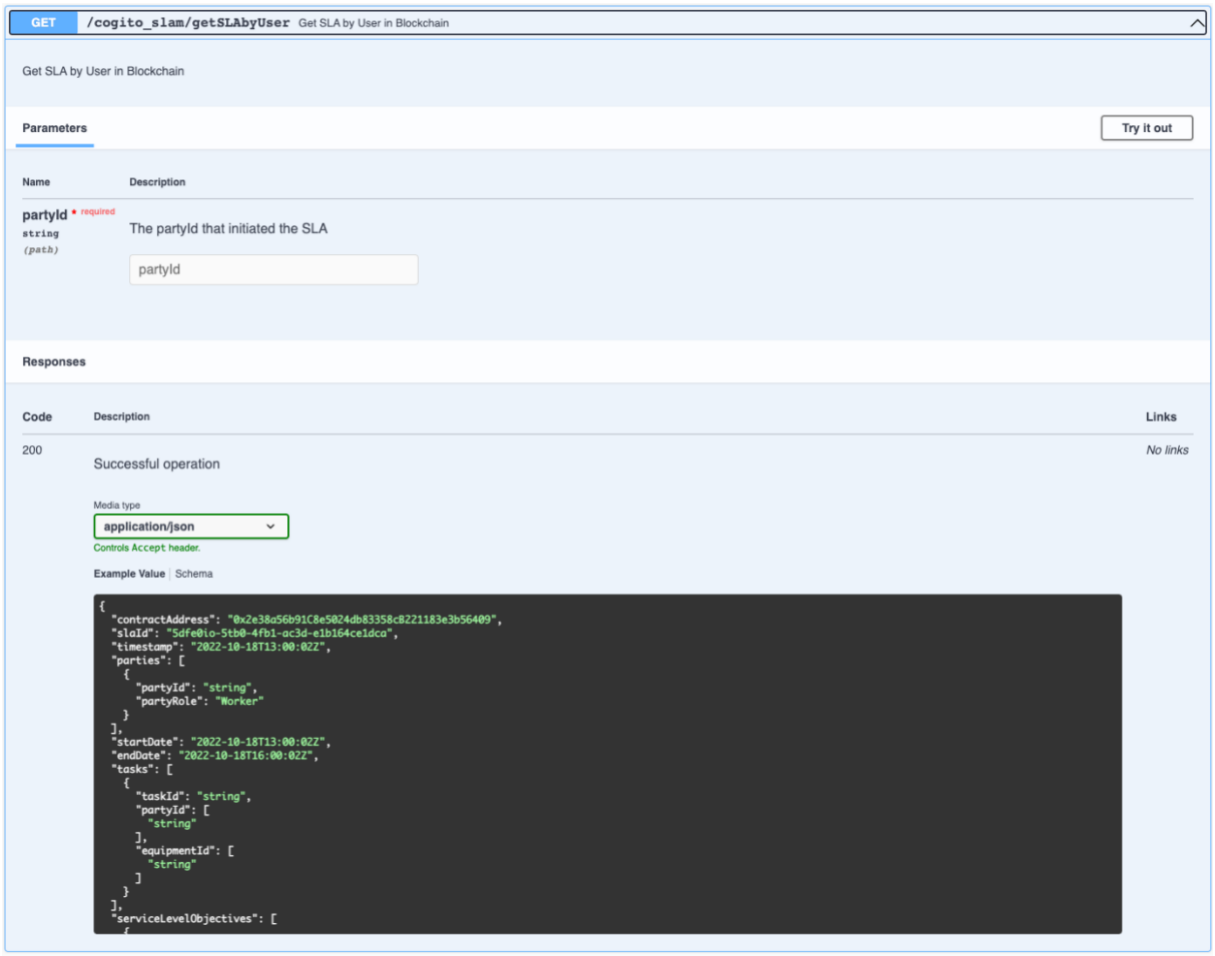


Figure 17 - The Swagger of getSLAbyUser

Then by selecting the contract ID, the user can further interact with the specific contract. By using the POST command with the **updateKPIbyContract** can update the KPIs for specific tasks. In the example presented on Figure 18, the user with party ID "7", interacts with the contract "5dfe0io-5tb0-4fb1-ac3d-e1b164ce1dca", and updates the "Percentage of Completed Work KPI" to 70% for task with ID "11". Then the user receives a success message and the KPI is also updated on the BC-SC platform. The Swagger of the service is presented on Figure 19.

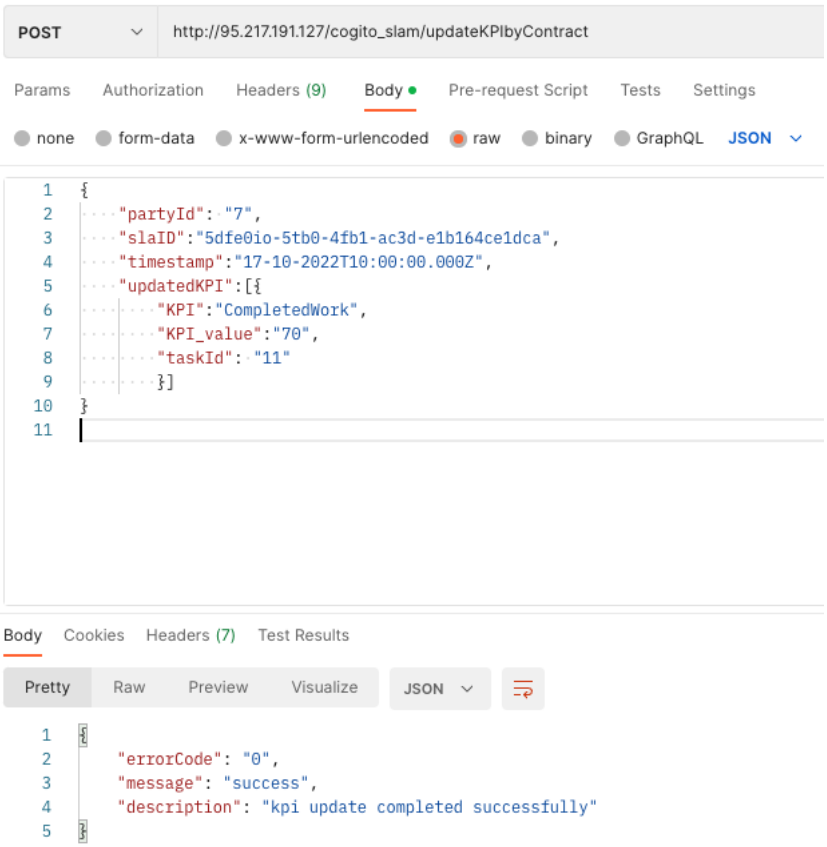


Figure 18 - Example of the updateKPIbyContract service using Postman.

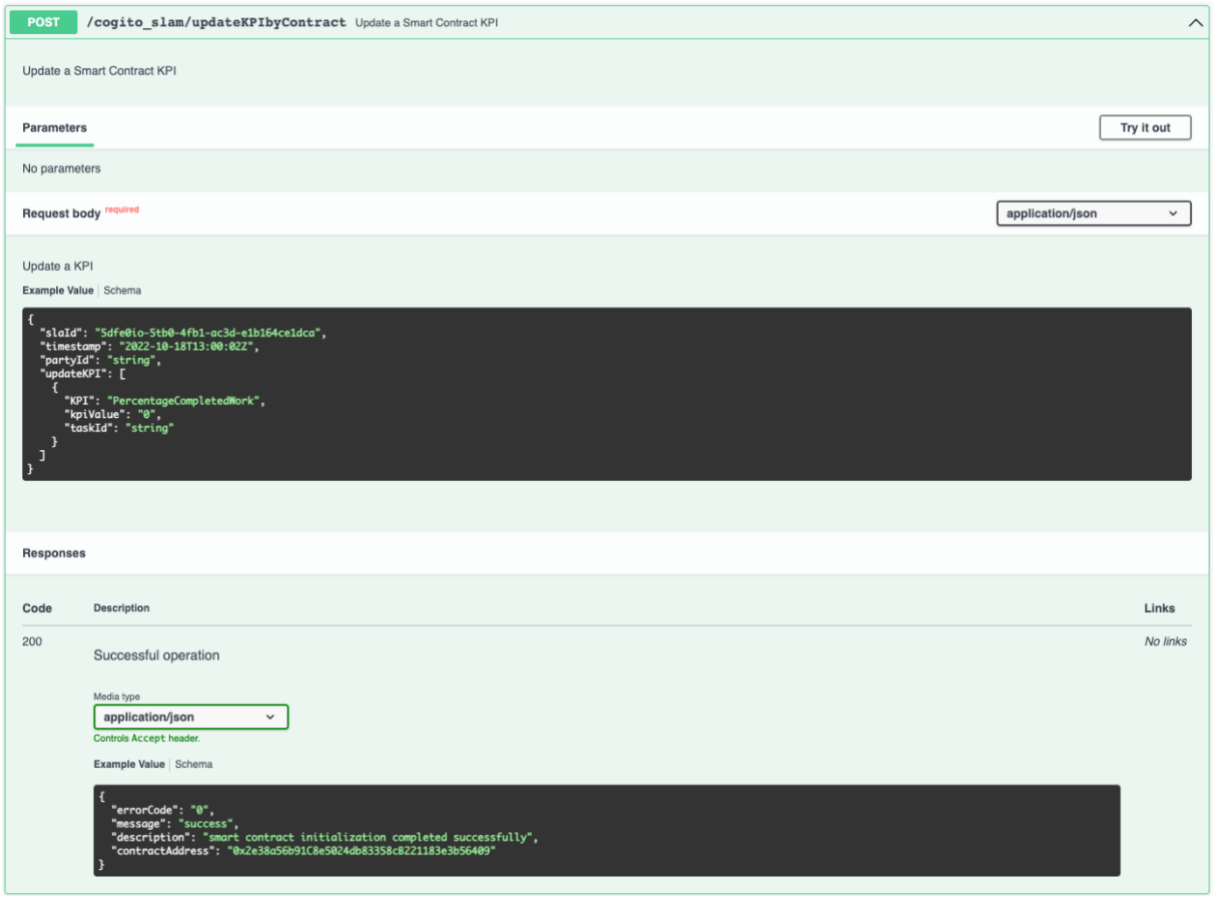


Figure 19 - The Swagger of updateKPIbyContract service.

Only the users that are registered with the specific tasks and the specific contracts can update the KPIs. However, any registered and certified user of the WODM tool can view either the KPIs or the overall progress of a specific contract. On Figure 20, by using the contract ID, the task ID and the KPI name as the request parameters of the GET command using the **getKPIResult** service, then the current value of the KPI is returned. In this case, the KPI value that is returned in the body of the response message is 70, as the value that was registered in the previous step from the user with ID "7". The Swagger of the service is presented on Figure 21.

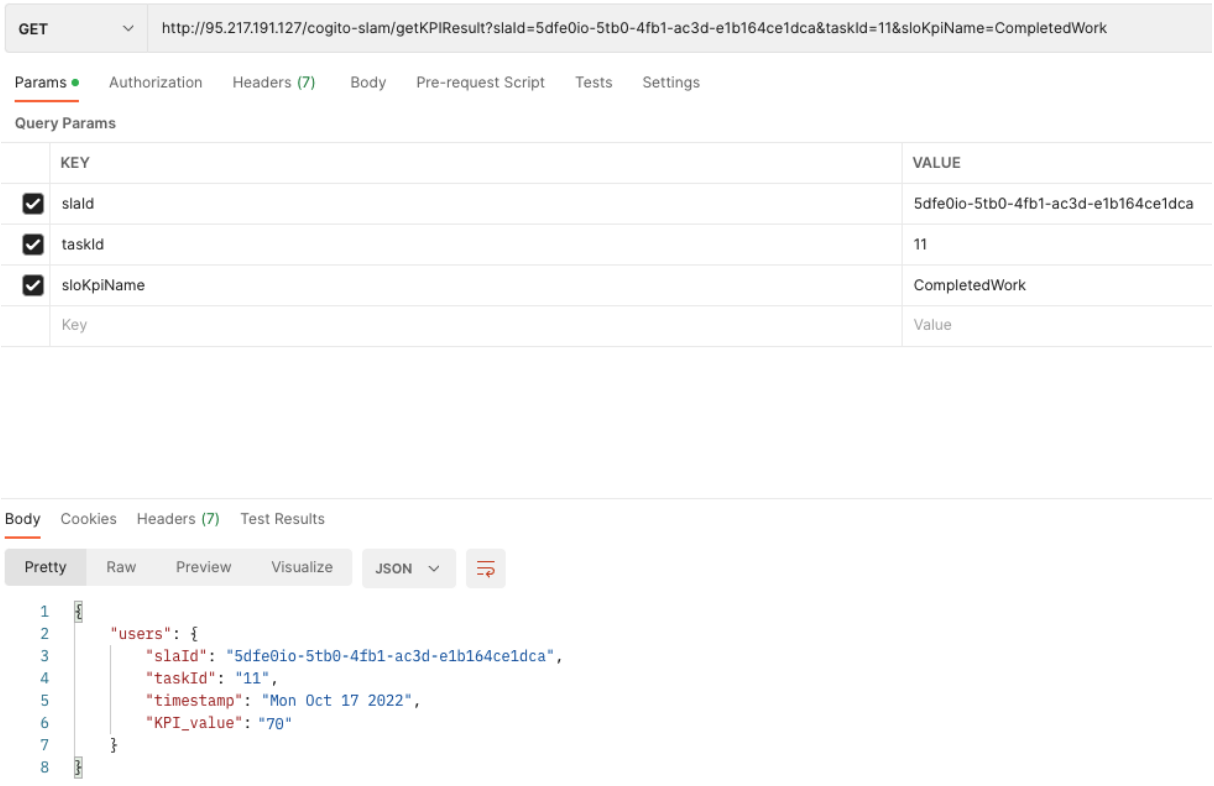


Figure 20 - Example of the getKPIResult service using Postman.

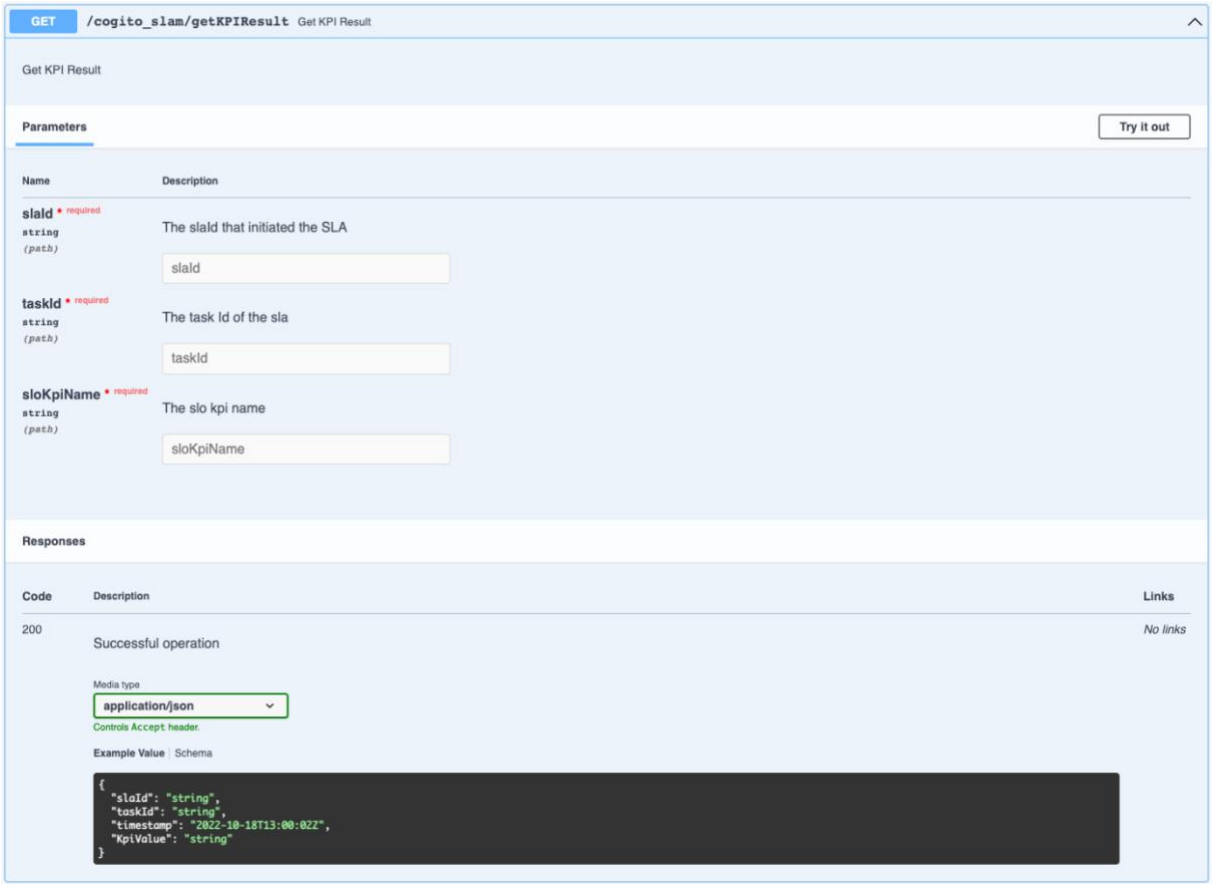


Figure 21 - The Swagger of getKPIResult service.

Finally, the overall performance of the contact is returned by using the **getSlaResult** service. The request parameters require the contract ID from the WODM user to reply with the overall performance of the specific contract. The overall performance of the contract is calculated from the formula in section 2.4.4. In this case, there are 17 tasks in the current work order, these tasks are all equally important since they have been assigned with a weighted factor equal to 1, only the task with ID "11" has the KPI "Percentage of completed work "at 70%, and therefore the overall performance of the SLA is 4.1%. The Postman output and the Swagger of the service is presented on Figure 22 and Figure 23 respectively.

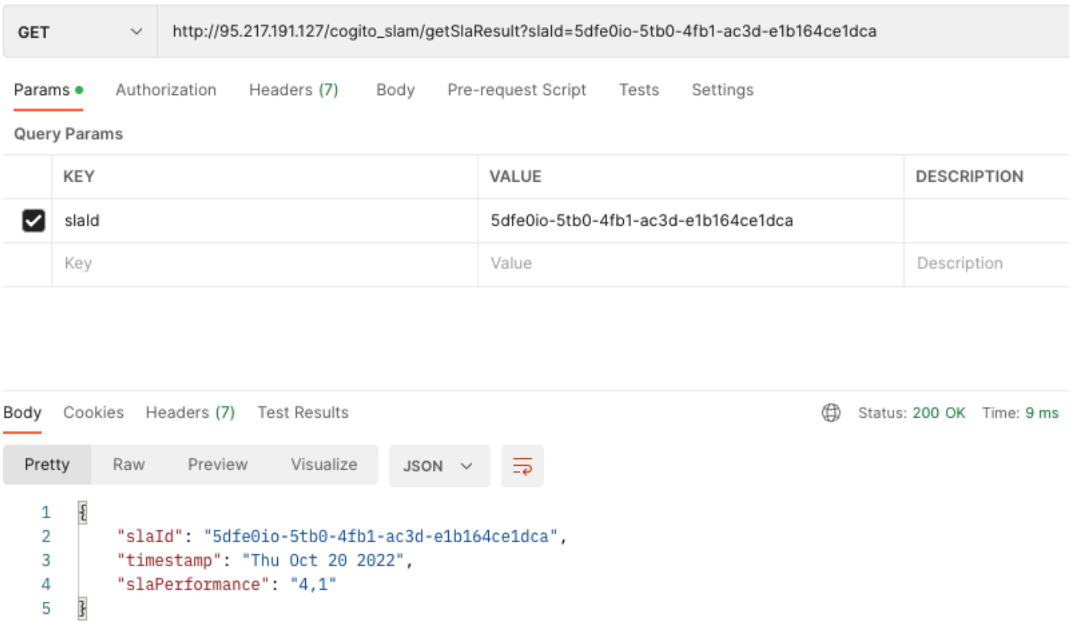


Figure 22 - Example of the getSlaResult service using Postman.

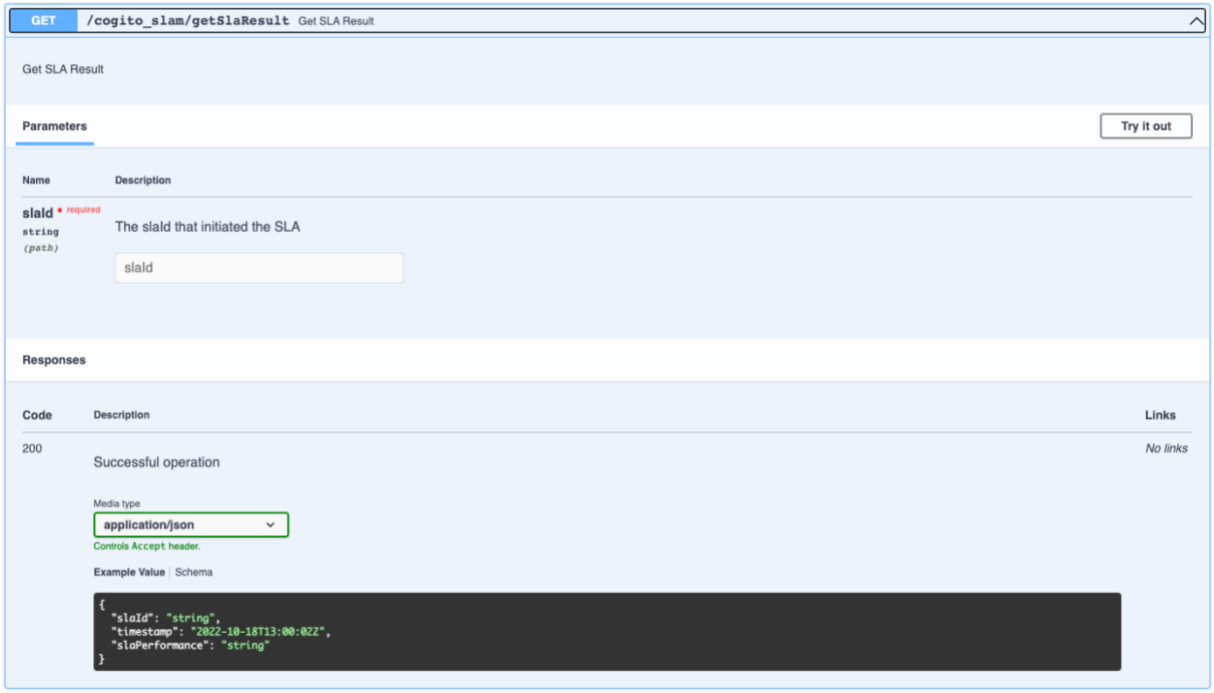


Figure 23 - The Swagger of getSlaResult service.

3.7 Development and integration status

The SLAM and BC-SC components presented in this document are fully developed and have all the functionalities that are described in UC1.1 and UC1.2. However, since the WODM component is significantly delayed no testing could take place. Therefore, the integration part was replaced by manual exchange of information based on the Postman software. As a conclusion, further testing is needed, and minor changes might be required when fully integrated.

3.8 Requirements Coverage

The current deliverable has created a solid framework for the design of SCs, that will provide the trusted means for task completion verification having as a goal to reduce the administrative overhead. As a data and transaction governance tool, it will act as a single source of truth that will increase trust between the involved stakeholders. It steps on the findings of T6.2 (Adaptive Processes/Workflow Modelling and Simulation-based Optimization) and D2.4 (COGITO System Architecture v1) to place the SCs and the associated Blockchain plugin alongside the rest of the COGITO components.

A set of KPIs was proposed that will sufficiently cover all the aspects of a construction process in a quantifiable and data driven manner. Moreover, this document provides the methodology framework for the usual paper-based SLAs, that are derived from conventional work orders, to be transformed into a fully functional SC. To accomplish the SC data model alongside the individual entities and their attributes was described in section 2.4.3. Additionally, API services and the associated data payloads were designed to serve this purpose.

However, the selected pilot sites for the COGITO project are characterized by a high degree of complexity and present many challenges. These factors are critical and therefore, the SLAM and BC-SC tools might require adaptations, or fine tuning, to meet their specific needs. This requirement is evident from the early stages of the project and is more prominent as tools are being developed and integrated in a complete solution. This is a continuous and ongoing process to fully record and analyze the exact needs and specificities of each task. Consequently, the SLAM and BC-SC tools will have to be refined at a later stage to fully address the pilot demonstration needs. The outcomes of the aforementioned refinements will be documented and reported in “D8.2-Integrated COGITO system v2” and “D8.3-Integrated COGITO system v3,” planned to be released in M30 (April 2023) and M34 (October 2023), respectively.

3.9 Assumptions and Restrictions

The proposed solution introduces a Blockchain based and smart contract enabled data governance scheme that offers security, transparency, and trustworthiness to the complex and fragmented construction industry environment. The already known problems of Blockchain technology that have been limiting large scale industrial and commercial applications, such as a scalability and transaction speed, have been successfully resolved by introducing new frameworks. For that reason, the Cosmos SDK and the Tendermint have been used to lay the foundations of the current solution. Due to their popularity are now considered mature solutions and are implemented in a wide range of applications.

However, since Blockchain technology has still limitations on the volume of data that can be stored, the current solution is designed to store only the minimum amount of information that is necessary as it is also presented in section 3.6. This does not include all the information that is used for work orders or the implementation of BIM, but only information regarding SLAs, KPIs and the involved stakeholders. The second assumption revolves around the familiarity of the average user with Blockchain technology. It is true that even tech-savvy construction stakeholders and highly digitized sectors are hesitant to adopt it as a mainstream solution. Therefore, in the concept of COGITO solution it will be seamlessly integrated with the WODM tool, so that the users will gain the benefits of security and transparency, but through a process that are already familiar.

4 Conclusions

The COGITO project aims to provide a SC platform that will provide the trusted means for construction task completion verification, having as a goal to reduce the administrative overhead. It will achieve that by introducing an efficient, flexible and scalable solution that will be seamlessly integrated into current workflow processes. The components presented here are based on Blockchain and SC technologies and together they form a data and transaction governance tool, that acts as a single source of truth.

The SLAM and BC-SC components work in close cooperation with the WODM tool, are based on the Tendermint BFT consensus engine, and the Cosmos SDK framework. These frameworks were selected since on one hand they offer security and reliability to the underlying consensus and networking engine, while they allow for modularity, scalability and interoperability on the application level. A number of KPIs that will be used to evaluate the work progress were identified and they will constitute the foundation of the SC KPIs. As a first step to deliver a concrete SC framework the data structure was provided. This includes 7 data entities, namely SLA, SLO, Task, Party, Rules & Actions, and Equipment. The various attributes of those entities were identified and thoroughly analysed, alongside with their relationships. The dedicated APIs for communication with other components were also developed and tested with an SLA JSON sample file.

References

[1] J. Ellul, J. Galea, M. Ganado, S. Mccarthy and G. J. Pace, "Regulating Blockchain, DLT and Smart Contracts: a technology regulator's perspective," *ERA Forum,* no. 21, p. 209–220, 2020.

[2] S. Johannes, B. H. Ulrich, F. Gilbert and K. Robert, "The Energy Consumption of Blockchain Technology: Beyond Myth," *Business & Information Systems Engineering,* vol. 62, p. 599–608, 2020.

[3] P. Kochovski, V. Stankovski, S. Gec, F. Faticanti, M. Savi, D. Siracusa and S. Kum, "Smart Contracts for Service-Level Agreements in Edge-to-Cloud Computing," *Journal of Grid Computing,* vol. 18, p. 673–690, 2020.

[4] J. Li, D. Greenwood and M. Kassem, "Blockchain in the built environment and construction industry: A systematic review, conceptual models and practical use cases," *Automation in Construction,* vol. 102, pp. 288-307, 2019.

[5] J. Mason, "Intelligent Contracts and the Construction Industry," *Journal of Legal Affairs and Dispute Resolution in Engineering and Construction,* vol. 9, no. 3, 2017.

[6] W. Nowiński and M. Kozma, "How Can Blockchain Technology Disrupt the Existing Business Models?," *International Entrepreneurship: New Perspective in IB Research,* vol. 5, no. 3, 2017.

[7] R. Zolin, P. J. Hinds, R. Fruchter and R. E. Levitt, "Interpersonal trust in cross-functional, geographically distributed work: A longitudinal study," *Information and Organization,,* vol. 14, no. 1, pp. 1-26, 2004.

[8] P. Pishdad-Bozorgi and Y. J. Beliveau, "Symbiotic Relationships between Integrated Project Delivery (IPD) and Trust," *International Journal of Construction Education and Research,* vol. 12, no. 3, 2016.

[9] S. Ahmadisheykhsarmast and R. Sonmez, "A smart contract system for security of payment of construction contracts," *Automation in Construction,* vol. 120, 2020.

[10] K. Sigalov, Y. X., M. König, P. Hagedorn, F. Blum, B. Severin, M. Hettmer, P. Hückinghaus, J. Wölkerling and D. Groß, "Automated Payment and Contract Management in the Construction Industry by Integrating Building Information Modeling and Blockchain-Based Smart Contracts," *Applied Sciences,* vol. 11, no. 16, 2021.

[11] J. J.Hunhevicz and D. M.Hall, "Do you need a blockchain in construction? Use case categories and decision framework for DLT design options," *Advanced Engineering Informatics,* vol. 45, 2020.

[12] M. Das, X. Tao, Y. Liu and J. C. Cheng, "A blockchain-based integrated document management framework for construction applications," *Automation in Construction,* vol. 133, 2022.

[13] E. C. S. Observatory, "Late Payment in the Construction Sector," ECSO, Brussels, Belgium, 2020.

[14] J. Li, M. Kassem, A. Ciribini and M. Bolpagni, "A Proposed Approach Integrating DLT, BIM, IoT and Smart Contracts: Demonstration Using a Simulated Installation Task," in *International Conference on Smart Infrastructure and Construction 2019*, 2019.

[15] Ž. Turk and R. Klinc, "Potentials of Blockchain Technology for Construction Management," *Procedia Engineering,* vol. 196, pp. 638-645, 2017.

[16] M. S. Kiu, F. C. Chia and P. F. Wong, “Exploring the potentials of blockchain application in construction industry: a systematic review,” *International Journal of Construction Management,* 2020.

ANNEX 1

Prerequisite tools:

- Go

Evmos is built using Go (opens new window) version 1.17.5+

```
go version
```

- Evmos binary is hosted in a Github repository and can be installed using git:

```
git clone https://github.com/tharsis/evmos.git
cd evmos
make install
```

- Truffle

```
Initialize the Truffle suite with:

truffle init
Deploy contract:

truffle migrate --network development
```

- Docker (alternative deployment):

```
make build-docker
```

To run evmosd in the container:

```
docker run -it -p 26657:26657 -p 26656:26656 -v ~/.evmosd/:/root/.evmosd tharsishq/evmos:latest evmosd version

# To initialize
# docker run -it -p 26657:26657 -p 26656:26656 -v ~/.evmosd/:/root/.evmosd tharsishq/evmos:latest evmosd init test-chain --chain-id test_9000-2

# To run
# docker run -it -p 26657:26657 -p 26656:26656 -v ~/.evmosd/:/root/.evmosd tharsishq/evmos:latest evmosd start
```

Single node installation

Script for customized local testnet, values can be customized for users' convenience:

```
# customize the name of your key, the chain-id, moniker of the node, keyring backend, and log level
KEY="mykey"
CHAINID="evmos_9000-2"
MONIKER="localtestnet"
KEYRING="test"
LOGLEVEL="info"
```

```
# Allocate genesis accounts (cosmos formatted addresses)
evmosd add-genesis-account $KEY 100000000000000000000000000aevmos --keyring-backend $KEYRING

# Sign genesis transaction
evmosd gentx $KEY 10000000000000000000aevmos --keyring-backend $KEYRING --chain-id $CHAINID
```

Local chain will start using:

```
init.sh
```

Before starting the chain, the state should be populated with at least one account using the keyring:

```
evmosd keys add my_validator --keyring-backend=test

#add aevmos tokens in chain's genesis file
evmosd add-genesis-account my_validator 10000000000aevmos --keyring-backend test

#add validator to chain
# Create a gentx
# NOTE: this command lets you set the number of coins.
# Make sure this account has some coins with the genesis.app_state.staking.params.bond_denom denom
evmosd add-genesis-account my_validator 1000000000stake,10000000000aevmos
```

Start the node:

```
evmosd start
```

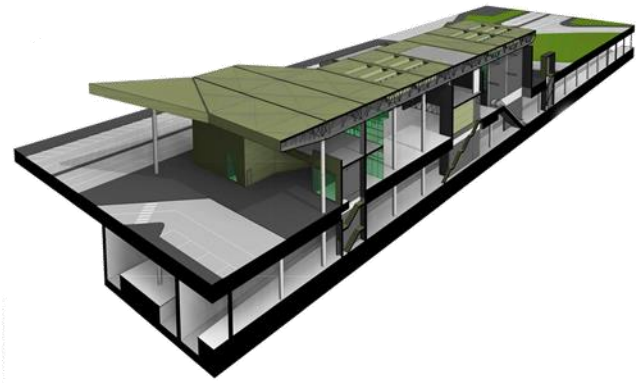
Key management

Key/mnemonic generation with:

```
evmosd keys add $KEY
```

To use key for Metamask, it can be extracted with the following:

```
evmosd keys unsafe-export-eth-key $KEY
```



COGITO

CONSTRUCTION PHASE DIGITAL TWIN MODEL

cogito-project.eu

HYPERTECH energy labs

UCL

AARHUS UNIVERSITET

THE UNIVERSITY of EDINBURGH

BoC www.boc-group.com

iti

POLITÉCNICA

QUE TECHNOLOGIES

Novitech NEW INFORMATION TECHNOLOGIES

ASM

ferrovial construction

ΟΛΥΜΠΙΑ ΟΔΟΣ

RHOMBERG SERSA RAIL GROUP

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 958310