Received: 1 May 2021 | Revised: 21 November 2021 | Accepted: 13 January 2022
DOI: 10.1002/cpe.6895

SPECIAL ISSUE PAPER

WILEY

Multi-party smart contract for an AI services ecosystem: An application to smart construction

Sandi Gec | **Petar Kochovski** | **Dejan Lavbič** | **Vlado Stankovski**

Faculty of Computer and Information Science, University of Ljubljana, Ljubljana, Slovenia

Correspondence
Vlado Stankovski, Faculty of Computer and Information Science, University of Ljubljana, Večna pot 113, Ljubljana, Slovenia.
Email: vlado.stankovski@fri.uni-lj.si

Funding information
European Union's Horizon 2020 Research and Innovation Programme, Decentralised technologies for orchestrated Cloud-to-Edge intelligence, Grant/Award Number: 815141; European Union's Horizon 2020 Research an Innovation Programme, Trusted, traceable and transparent ontological knowledge on blockchain, Grant/Award Number: 957338

Abstract

Various smart applications, such as in the domain of smart construction, require the use of artificial intelligence (AI) based services. In order to support such environments, various AI/knowledge service provider and consumer ecosystems have started to emerge. Within such ecosystems, the goals are to improve the quality, reliability, dependability in terms of governance and trust in the data which is exchanged among the various actors, which must be supported by specific business models. This work introduces a novel multi-party smart contract (SC) that is designed to address the above mentioned goals. Specific service level agreements that illustrate the interactions among the AI service providers and consumers are also presented. The developed multi-party SC supports different pricing schemes, which are analyzed in detail against the goals of the study.

KEYWORDS

artificial intelligence service, blockchain, service level agreement, smart contract

1 | INTRODUCTION

Construction projects involve many complex problems falling under the broad categories construction-site management, safety at work, resources management, waste and assets, building progress monitoring, and early (disaster) warning. Our previous work described different building use cases that are addressed in the DECENTER[1] project, particularly those involving the Internet of Things (IoT), artificial intelligence (AI), and cloud technologies. These software services are built using virtualized computing resources offered by cloud providers, software, such as web servers, libraries and micro-services, various digital content and the ability to align all the collected information with the building information model (BIM).[2,3] Various mechanisms are needed that facilitate the resource allocation in the context of monetization of the contributed added-value in such a business collaboration. Moreover, it is also necessary to assure dependability, reliability, security, quality of service (QoS), and other software properties, which represent service level objectives (SLOs) of various service level agreements (SLAs).

The goal of this study is to address the requirements of the software engineering community for flexible and highly automated collaborative provisioning and the use of computing resources and software services based on SLAs. This work concentrates on the analysis and implementation of various semi-automated monetization approaches with SCs on a blockchain layer, which may underpin the software decentralized finance (DeFi) aspect in the near future. The idea is to stimulate the collaborative provisioning of high-quality, dependable software services to the end users, which form part of specific SLOs. Examples of the implemented possibilities for flexible SLA offers include a fixed and variable pricing model, the time-limited and quota-based provisioning of software services, constraints-based access to services, revenue sharing in multi-party stakeholder scenarios, and other mechanisms making it possible for an engineer to deliver software services with sufficient business flexibility. This approach requires the development and evaluation of a new architecture that can be used to automate the process of software-services provisioning and consumption by using flexible SLAs.

The present approach relies on recent trends in the DeFi domain. Traditional currency transactions among people and companies are often facilitated with a central entity and controlled by third-party organizations such as banks. Bitcoin as the first decentralized digital currency was

presented and launched as an alternative to centralized banking solutions. It is based on blockchain technology and has many benefits.[4] Through the adoption of bitcoin by the world's population, blockchain technology has led to opportunities in many other areas, such as IoT, cloud computing and software engineering in general. These developments have led to the launch of other dedicated cryptocurrencies—Ethereum[5] as a SC ledger, IoTa[6] as an IoT-based ledger, Ripple[7] as a transaction cost optimization ledger, ledgers for anonymous transactions, protocols and other blockchain systems covering a plethora of use cases. As an example from the cloud domain, the distributed storage system StorJ[8] is an alternative to commercial storage solutions such as Amazon S3 storage, Google Drive, or Dropbox, and provides the same types of services with high encryption security and full transparency.

In this article we propose a new architecture for flexible software-services provisioning through SLAs using SCs, which is demonstrated in an example of a AI service for an AI video component (VC) application. The AI application is implemented as a container image (CI). We exploit the benefits of blockchain as a critical component that differs from other existing solutions. In this context, SCs form part of the new architecture. They are used to fulfill the functional requirements, such as the necessary agreements among the user and system roles, which address not only monetization but also QoS assurance requirements. By developing solutions for several monetization use cases, we aim to cover a potentially large number of stakeholders in accordance with their usage preferences, such as regular, occasional, demanding users. Hence, this new architecture is designed, having in mind the requirements for dependability, transparency and various pricing-scheme approaches through the use of SCs. By implementing the AI application on top of blockchain, it is possible to empirically compare this architecture with traditional monetization approaches in quantitative and qualitative terms. The contributions of this article are: (i) a novel architecture for flexible software services for smart construction, and (ii) the introduction of multi-party SC templates that support different SLA pricing schemes.

The common point of every building project is the unique environment, including stakeholders, that would benefit the DECENTER platform to facilitate their process. These stakeholders strive to use application services optimally where it would be necessary to plan the smart application(s) along with the overall planning activities for the building. Due to this requirement, our research activities also involve architects and construction-site managers, which is elaborated further in the following sections.

The rest of the article is organized as follows. Section 2 covers the background to the area. Section 3positions our work in the context of other related works. Section 4 depicts the use cases, stakeholders, and requirements. Section 5 presents the blockchain enabling DECENTER architecture and its implementation. Section *6* describes the implementation of the pricing schemes, including the design of the SCs, with a focus on the consensus pricing scheme with the workflow explained in detail. Section 7presents the evaluation methodology and the quantitative results measured using the testnet environment of the Ethereum network. Section 8 explains the significance of the obtained results. Section 9 draws the conclusions.

2 | BACKGROUND

In 1991 Haber et al.[9] presented the theoretical idea for the blockchain (originally named block chain) concept, on how to certify digital documents to assure tamper-proof data integrity. Blockchain is a continuously (time-based) growing, immutable list of records, called blocks, which are linked and secured using cryptography mechanisms. Each record ordinarily contains metadata—a reference (cryptographic hash) to the previous block, timestamp, and transaction data. Critical properties of blockchain are elevated distribution, no central authority, irreversibility, accessibility, time-stamping, and cryptography. There are several mechanisms that help build distributed systems with such properties, which are discussed in detail elsewhere.

The first practical attempt at blockchain technology was presented with the launch of the cryptocurrency bitcoin[10] in 2009. Bitcoin's simplicity, the simple sending and receiving of digital assets, encouraged many researchers and blockchain enthusiasts to launch their blockchain cryptocurrencies. Vitalik et al.[11] presented an interesting concept with the introduction of Turing complete smart contracts (SCs) that are similar to general (notary) contracts with limited, but at the same time sufficient, functionalities to cover several different use cases. SCs can be viewed as an expression of an SLA, and represent a protocol intended to digitally facilitate, verify, and enforce the negotiation or performance by implementing arbitrary rules. It contains exactly one constructor, functions, variables, events, and similar entities, which are used in the programming.

The Ethereum network uses the built-in, fully fledged, Turing-complete programming language solidity, which is used in this study to design and implement SCs. The lifecycle of an SC is illustrated by following an example involving several actors.[12] A developer designs a dedicated SC or uses an existing one. Then, the SC template is deployed on *an address* on the production (mainnet) or development (testnet) network. Upon deployment, the SC invokes the constructor only once, while the address owner usually has higher privileges, for example, it can destroy the deployed SC instance. Other participants, *public or determined addresses*, can trigger an SC instance through the supported functions that lead to the sending of another transaction, triggering another SC. Theoretically, *ad infinitum* as illustrated in Figure 1.

Blockchain and SCs are technologies underpinning the design and implementation of our new architecture, on top of which an AI service is provisioned. To understand the monetization in the domain of cloud services we first need to understand the economic ecosystem of resources and services deployed in the cloud. In the first place, each cloud provider must consider the total cost of ownership for the on-premises software by considering the cost of the resources, equipment, computing and networking infrastructures and the lifespan of the services. All of these aspects can be described as the operational and maintenance costs of the services running on demand.

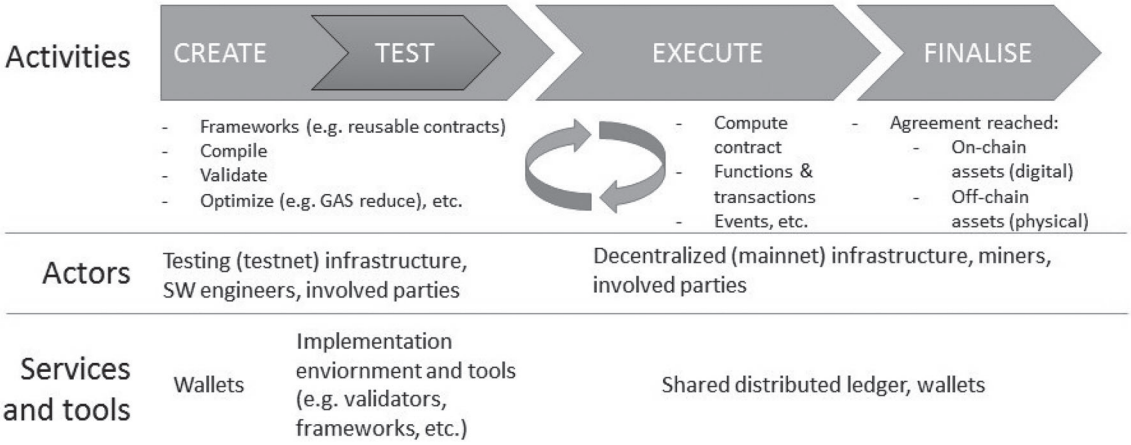

FIGURE 1 The lifecycle of a SC with an outline of activities, actors, services, and tools

Currently, there are various special-purpose monetization services, such as the YouTube channel monetization, mobile applications advertising monetization and similar, which commonly have relatively expensive operational costs. It may be possible to implement a flexible blockchain monetization overlay on top of cloud services, which would significantly reduce the operational costs.

When developing an SC, the software engineer has to define its publicly available functions, while at the same time address the concern of potential attacks due to the buggy code. Since the introduction of design patterns for SCs,[13] various tools have been developed for different SC development and analysis purposes. Known SC software tools or frameworks that were explored and used for this study include: (i)*Truffle Suite*,[14] which is an excellent development environment, and asset pipeline for blockchains using the Ethereum virtual machine (EVM), with the purpose to facilitate the development of SCs; (ii) Zhang et al.[15] presented *TOWN CRIER*, aiming to provide trustworthy data to SCs; (iii) Kosba et al.[16] developed *HAWK*, which is a decentralized SC system that enables developers to code privacy-preserving SCs; (iv) Luu et al.[17] presented *OYENTE*, a symbolic web tool, which discovers security bugs in Ethereum SCs; (v) Chen[18] presented *GasReducer*, a tool that detects anti-pattern instances in deploying and invoking SCs with recommendations to reduce the Ethereum GAS cost; (vi) *Tetrix*[19] is an open-source tool for lowering gas costs during solidity contract instantiation; (vii) *OpenZeppelin*[20] is a framework of reusable SCs for Ethereum and other (EVM and eWASM) blockchains that has been widely tested using many existing Ethereum tokens.

In this article we use some of these tools to implement several SCs supporting different pricing schemes to provide AI-based services. The new solution is an architectural overlay that addresses the requirements of different types of end users interested in the software-services economy. The developed architecture and its implementation are explained in detail in the following section. Following that, we proceed by elaborating the specific SCs that were developed in the course of the study.

3 | RELATED WORK

Blockchain has monetization potential, which has been demonstrated with practical applications and the latest global trends.[21] The application areas with the most potential are digital assets registry management, solving the issue of billions of "unbanked" people, long-tail personalized economic services and payment channels concerning the creation of financial contracts executed over time. Yoo[22] analyzed the potential of blockchain technology from the global financial perspective and thus also leaning for the use case of micropayments. Ankenbrand et al.[23] presented a comprehensive comparison between the current monetary system and blockchain. Haiss et al.[24] outlined the actual operational and regulatory challenges regarding scalability, interoperability, standards, governance and others, which should be defined by governments and financial institutes. All of these works explore ideas from a high-level perspective and do not provide a concrete monetization approach for software services, architecture or use cases, which is the goal of the present study.

Cloud computing is already widely adopted in all information and communication technology (ICT) environments and domains such as civil engineering, for its efficiency, availability and hardware resource scalability. Cloud application architectures usually consist of a variety of sub-systems, such as front-end and back-end platforms, cloud-based delivery and networks that address specific end-user needs and requirements. Currently, significant efforts are invested in achieving a high QoS, which must be maintained for any successful business scenario. To address the QoS problem, some studies have focused on the design of cloud-based orchestration systems that provide the intelligent delivery of CI-based applications across multiple cloud providers, such as a file uploading and a video conferencing application.[25] Efthymiopoulou et al.[26] have proposed a live video-streaming architecture with a focus on dynamic resource allocation of the infrastructure. This application does not consider

TABLE 1 Smart application approaches in cloud-computing environments

Study	Computing environment	Blockchain, smart contract	Implementation	Smart contract advanced features	Application areas
Hu et al.[28]	Cloud	Yes, Yes	No	Multiple stakeholders	Smart construction
Paščinski et al.[25]	Cloud	No, No	Yes	/	Video-conferencing, file upload
Efthymiopoulou et al.[26]	Cloud	No, No	Partial	/	Video-streaming
Al-Turjman[27]	Cloud, edge	No, No	Yes	/	IoT
Lu et al.[29]	Cloud	Yes, Yes	Yes	Multiple stakeholders	Product Traceability
Xia et al.[30]	Cloud	Yes, Yes	Partial	Multiple stakeholders	Medical data sharing
Kochovski et al.[31]	Fog, edge, cloud	Yes, Yes	Partial	Smart oracles	Smart construction
Present approach	Fog, edge, cloud	Yes, Yes	Yes	Multiple stakeholders, consensus, governance, block-time triggered events	AI/knowledge services in smart construction

geolocation-based deployment of the software services to improve connectivity, such as latency or bandwidth. An adaptive routing approach, as proposed by Al-Turjman,[27] has also been proposed for cost-effective video streaming. The present study, however, is the first to investigate the needs for monetization as well as the maintenance of a high QoS in the cloud-services economy, which employs blockchain and SCs.

Our study aims to increase the overall system reliability, dependability and robustness, when various software services are developed, engineered, deployed, and operated in a highly distributed way, across multiple cloud providers, that is, across what is known as the edge-fog-cloud computing continuum. The research idea of integrating SCs with existing AI services has been identified as a promising research direction.[28] Several highly focused studies have used blockchain technology to improve the software product traceability and quality preservation, such as the study of Lu et al..[29] The work of Xia et al.[30] focused on the adoption of blockchain for trustless medical data sharing. In our earlier work we proposed a new methodology for applying fog computing to achieve smart construction sites,[31] but neglecting the integration of actual SCs. However, the overall performance of such systems has not been addressed sufficiently, and not in the context of SCs. An attempt to integrate SCs in an existing cloud system for VMI and CI management was presented in our Horizon 2020 ENTICE project[32] as an agreement-management component for the users of a distributed VMI and CI repository. The work presented in this article uses the various benefits of SCs to define, develop, and test various monetization strategies, which may be useful in the domain of the cloud-services economy. Lately, there are many DeFi platforms emerging that focus on a P2P lending-and-borrowing use case (e.g., Venus protocol[33] and many others).

The novelty of this study is an architecture that applies the possibilities offered by SCs in the domain of software engineering and might stimulate flexible business collaborations when dynamically delivering software services in the area of smart construction. This new architecture is demonstrated and tested on construction domain use cases, and the cost of such an operation is presented and compared for the various newly designed SCs. Moreover, Table 1 aligns the present approach with other related work according to some key aspects relevant for cloud-computing platforms.

4 | USE CASES

To achieve information integration for various construction-process-related requirements, it is necessary to develop a methodology that would turn a specific construction site into a smart-and-safe construction site. This includes the setup of a specific number of cameras and sensors on the ground, including sensors wearable by the people on the site, such as the construction workers, visitors, site managers, and on or near specific objects, such as the building equipment, materials, waste and so forth. The DECENTER Fog Computing and Brokerage Platform makes it possible to apply various AI methods on video streams to detect specific details and objects and use the information to issue notifications, for example, to the construction-site engineers and managers.

4.1 | Use cases

We explored mechanisms for information gathering, fusion, and enrichment that can provide intelligence during the construction process and help improve various aspects of the work. The collection of information related to the ongoing construction process can be useful for both time-critical operations, such as collision detection and early-disaster warning, as well as longer-term logistic and other operations, such as ordering material, documenting the process. We initially considered the following use cases:

1. Improve safety at work by issuing notifications to the construction-site manager. For each worker at the construction site we will determine whether she/he wears a helmet, a working suit or safety vest and his/her location according to specific zones. In the case of a safety concern, the supervisor will be notified. With this scenario we plan to demonstrate DECENTER's capability to provide high QoS and fast response times to the smart applications.
2. Construction-site surveillance for any vehicle that appears at the entrance to enter or leave the construction site. The vehicle can be identified by type, color, and the number on the license plate. Any person who appears at the entrance to enter or leave the construction site can be identified as a worker or a visitor. This scenarios rely on DECENTER's capabilities:
   - to select an AI method/model from a repository that is required and optimized for specific computational resources,
   - to use SCs to access the AI models to support privacy preservation through data-movement-related regulations and the certification of cloud-to-edge providers for security and reliability. These are achieved using specially designed SCs and smart oracles.
3. Management of resources, waste, and assets is a scenario that requires continuous construction-site monitoring to identify various objects and provide possibilities to track them. For example, in the case of any changes in quantities, the construction-site management is notified. Additional possibilities include equipment/tool detection, monitoring the quantity of waste, stock/assets on the site and similar. This scenario demonstrates DECENTER's capability to integrate various AI methods, which would significantly increase the possibility to monitor resources, machinery, inventories, and material consumption.
4. Working conditions is a scenario that involves monitoring the environmental conditions of the site, such as temperature, wind, noise and a comparison between optimal, minimal, and actual working conditions. With this scenario we intended to help maintain an optimum working conditions together with environmental protection. The scenario basically demonstrates DECENTER's capability to use various IoT sensors along with camera-based video streams. This is due to its use of the SensiNact IoT platform[34] that is embedded in DECENTER.

This work further elaborates use case no. 2, which slightly complements use case no. 1, and has been selected for initial implementation.

4.2 | Actors and roles

The proposed use cases investigated might involve several actors and roles with different prior-knowledge of smart applications and services. Due to this, one part of the evaluation of the DECENTER fog computing platform is related to its utility when its users may be other domain experts.

A smart construction site is one equipped with cameras and sensors, potentially also computational resources, such as edge nodes, either leased or owned, that can be used to generate data and perform AI processing closer to the data sources. The implemented AI-based applications can issue construction-process-specific messages either via onsite signalization or notifications to mobile phones and similar. The following actors could potentially be involved:

- Construction investor/client, supervisor that may benefit from better construction-work tracking and is notified about progress or safety breaches,
- Construction company, the goal of which is to reduce operational costs and provide a safe working environment,
- Construction site manager is a person employed by the company that coordinates the work at the site,
- Worker(s) that may benefit from working under optimum conditions and from applications for safety at work,
- Construction equipment rental/lease providers support the construction site with equipment,
- Objects: hard hats, protective vests represent equipment that should be monitored,
- AI methods: detect and recognize people and objects, and decide when to issue specific notifications based on video streams and moving objects. Our initial list of AI methods and models can be greatly extended after an initial implementation,
- Data sources: cameras (e.g., IP or USB cameras) and sensors that are used to collect raw data.
- Any required computing infrastructure.

It is obvious that the actual smart construction-site setup is non-trivial due to the unique requirements within each construction project. However, DECENTER provides the ability to seamlessly integrate various cameras and IoT sensors due to its integration with SensiNact.[35] The setup process is even simpler with the introduction of multi-party SCs that enable the dynamic usage of AI services considering all the involved stakeholders.

4.3 | Requirements

The following is a summary of the key requirements that have to be addressed for the individual smart-and-safe construction use cases:

- QoS and response time of the application, since in various time-critical situations notifications must be delivered quickly,
- Switching AI methods on the same video stream so that specific information is gathered that can only be present in a certain video sequence,
- The ability to identify people at the construction site, which requires the use of specific privacy-preserving AI methods/models,
- Aggregating IoT sources that have to be positioned on the ground, for example, various sensors and cameras.

In the following we present the DECENTER Fog Computing Brokerage Platform that is specifically designed to address such requirements with AI service provision through SCs, and may serve an advanced Science.

5 | THE DECENTER FOG AND BROKERAGE PLATFORM

The DECENTER Fog Computing and Brokerage Platform integrates IoT, AI, blockchain, and cloud-computing technologies to facilitate the process of creating smart applications, where computing resources, that is, cloud-to-edge infrastructures and IoT resources, for example, sensors and actuators, can be orchestrated in dynamically created, federated environments.

The key innovations are the following:

- Fog computing platform to orchestrate cloud-to-edge resources, which provides QoS-aware orchestration of microservices for AI applications based on their QoS requirements;
- Blockchain-based framework for the regulated cross-border brokerage of resources; the present study uses Ethereum testnet.
- Smarter IoT fabric, based on AI algorithms for extracting context information from IoT data;
- Hybrid decentralized AI models exploiting, which addresses the needs for data locality and time-critical aspects of the big-data problem.

In our work we focus on the blockchain-based framework based on SCs that serves as a bridge between stakeholders and AI service providers and is described in detail in the following section.

5.1 | Main DECENTER ecosystem layers

As shown in Figure 2, DECENTER's platform is composed of four main layers: infrastructure, platform, application, and blockchain layers. The infrastructure layer is composed of hardware equipment such as: computing, network, storage, and IoT devices. The platform layer is composed of logical components for the discovery, brokerage, and orchestration of resources from across the cloud-to-edge continuum. The application layer is used to compose applications to exploit a variety of AI methods and models. The blockchain layer is composed of decentralized applications (DApps) provided by the SC-enabling blockchains such as Ethereum.

5.1.1 | Infrastructure layer

New AI applications are time-critical and thus they need to address specific QoS requirements, for example, low latency, high bandwidth, or fast computations. To address such requirements, DECENTER's infrastructure layer was designed to distribute the computing from the cloud, via the fog to the edge. Multiple computing tiers mostly differentiate among properties, such as computing performance, network performance, and geolocation.

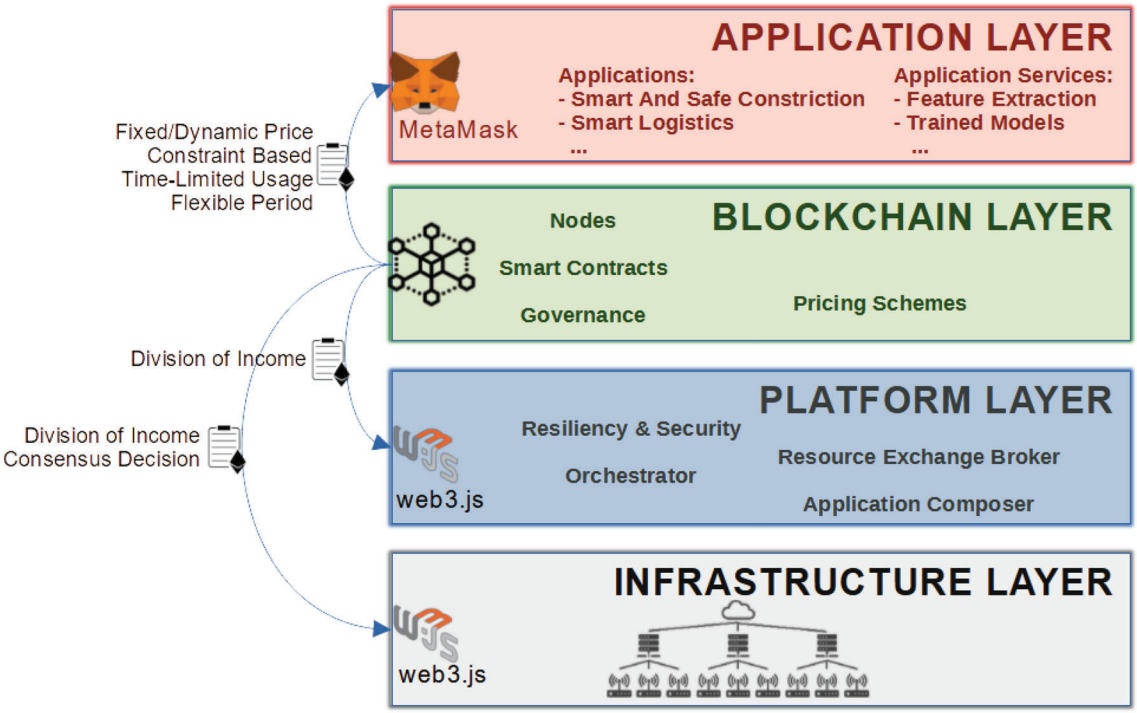

FIGURE 2 DECENTER's platform main layers with the focus on smart contract interactions among the layers

The cloud tier is composed of data-centers offering high computing power, large storage capacities, but lower network performance. This tier allows the utilization of private and public cloud services. In contrast to the cloud tier, the fog tier offers better network performance. The computing resources in the Fog tier are perceived as computing resources that exist in the middle between the cloud tier and the edge tier.[36] The infrastructures from this tier are connected to the cloud tier through WAN connections or through the Internet. The edge tier is a highly distributed computing environment, composed of low-power devices that make it possible to perform a simple computational task in close proximity to the physical devices, that is, cameras and sensors.

Since network connectivity is crucial, it is important to establish a fail over plan. On the construction site it is reasonable to establish multiple network gateways and mesh networking to minimize network failures, especially among the services that cannot communicate on premises. For example, there are services available in the fog or edge infrastructure or specific blockchain public nodes not present at the construction-site premises that can be accessed only through the public network. With mesh networking it is possible to maintain high network availability at the majority of construction sites.

Edge infrastructures are chosen only upon their network connections to the sources of data (i.e., sensors). In practice one sensor can be connected to only few edge nodes. An edge node is selected by using a multi-criteria decision-making process. This makes it possible to address low power, cost-effective requirements. The DECENTER platform envisions a scenario where infrastructure providers can interact with each other and federate their resources to provide services to users. However, DECENTER is not only about bringing advances to the infrastructure layer. It leverages on existing hardware to deliver fog-computing capabilities.

5.1.2 | Platform layer

The platform layer is placed between the infrastructure layer and the blockchain layer. The platform layer offers DECENTER multiple functionalities: (1) Setup and manage highly distributed and heterogeneous computing environments, (2) compose cloud applications from containerized microservices and define QoS requirements, (3) deployment and orchestration of applications, (4) brokerage and negotiation of resources that belong to different administrative domains. The fog computing platform, as a pillar sub-component, is responsible for application composition, resource allocation, and orchestration throughout the complete edge-to-cloud continuum. The application composition sub-layer formulates the applications in a formal TOSCA[37] specification and forwards it to the orchestrator. The orchestrator considers the current status of the available infrastructures, triggers the decision-making algorithms[38,39] and executes the orchestration process based upon the decision.

5.1.3 | Blockchain layer

The blockchain layer is placed between the platform layer and the application layer to enable blockchain technologies such as SCs among the layers. The main purpose of the layer is the brokerage platform as a mechanism that is responsible for managing the federation of fog-computing platforms across different administrative domains. In particular, the brokerage platform allows infrastructure providers to register their resources on the blockchain, whilst users can select a resource that will satisfy their application's QoS requirements. The process of registering and selecting resources is performed by executing the desired pricing-scheme-based blockchain SCs.[40] The interaction of SCs in the blockchain layer with other layers depends on the SCs' specifications and are depicted in Figure 2. The formulation of the blockchain layer, as an independent platform layer, derives from the potential emerged blockchain technology capabilities to support interoperability between different blockchain ledgers.

5.1.4 | Application layer

The application layer includes two types of entities that offer complementary functionalities: *Applications* and *Application Services*. Applications exploiting AI methods and models using the cloud-to-things resources for the execution of well-defined business tasks that are designed by application developers. On the other hand, application service provide some general purpose that is, not application-specific, functionalities that can be used by different applications. For instance, feature-extraction methods from video streams, for example, to detect workers wearing safety equipment, can be provided as application services. The video streams captured from the cameras are not stored in any kind of storage, because cameras are frequently moved around the construction site. In circumstances when the storage of video streams is mandatory, due to the needs on transparent auditing, it would be most reasonable to store the video streams encrypted on an InterPlanetary file system (IPFS) storage with the reference on the blockchain.

Application developers can make use of them to implement applications' business logic, without the need to implement their provided functionalities from scratch.

5.2 | High-level architecture

In the following we depict the process of setting up a smart construction site with some basic functionalities provided by DECENTER's platform through the multi-party SCs to highlight the high-level architecture.

5.2.1 | AI application design

In order to construct a specific scenario for implementation, the architect has to imagine the whole construction process and various situations. Figure 3 depicts such situations, for example, vehicle identification, identifying people and using safety equipment, detection of harmful working conditions, identification of objects.

In this specific scenario two actors are present: camera 1, type: static camera, attached at a height of 3.0 m, upper corner, wide-angle lens, large depth of field, short 1/1000-s exposure, a minimum width of 20.0 m, Person 1: a construction worker carrying a protective helmet and a safety vest, and Person 2: a site visitor, normal civilian, who does not carry any protective equipment. The camera is located on the roof of the construction site.

The camera captures the image of the whole site. Scene 1 is without moving objects, without people or moving machines. Scene 2: a construction worker appears on the site. The worker is walking from the construction office in the direction towards the site and in the opposite direction. The construction worker wears a safety helmet and a safety vest. The scene lasts as long as the worker is captured in a frame. In Scene 3 a visitor appears. He is a visitor walking from the construction office towards the work site and vice versa. The visitor does not wear a safety helmet and safety vest. The scene lasts as long as the worker is captured in a frame.

The captured data, moving images, are analyzed using AI methods (e.g., TensorFlow[41]). The purpose of the recognition is to identify the use of certain prescribed safety equipment by people moving on construction sites. From the pictures we can see whether the person that appears on the construction site wears a protective helmet and a security vest. From the learned database, we identify whether the person that appears on the site is authorized to enter the construction site or not. In the case of the perception of an unauthorized movement a warning message is sent to the construction-site manager.

As shown in Figure 3, the smart application uses the following components of the DECENTER platform: orchestrator, message broker, computing infrastructures pool, and AI model repository. The backend is developed as an application-related service and blockchain is the public ledger. The architecture outlines the workflow of the *Fixed price monetization scheme* that involves the end-user on a construction site that purchases the AI CI

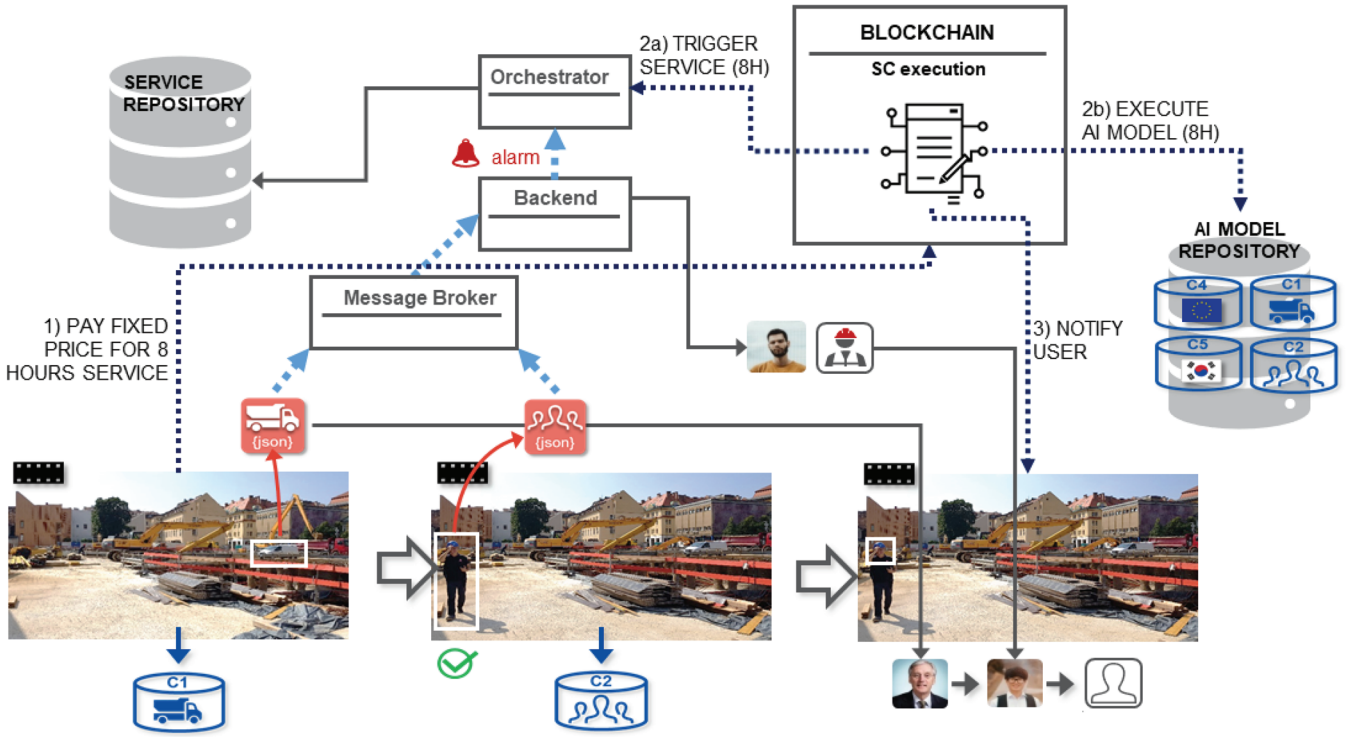

FIGURE 3 High-level architecture: Application design and components interaction on a fixed-price scheme

service (step 1) for a fixed period of time. After the successful payment through the SC, all the related services are deployed (step 2a and 2b). Finally, after the agreed time the AI CI services un-deploy and notify the end-user.

5.2.2 | Role of the blockchain components

The system components and end users can interact with the public blockchain, in particular with Ethereum SCs, through the designed services enabling Web3 and through the web-based GUI. The latter is the main entry point for the users who want to establish SLA preferences based on the QoS requirements and the proffered pricing scheme. The web GUI connectivity with the Ethereum ecosystem is assured using the Ethereum bridge Metamask,[42] which is a browser add-on. The users interact with the DECENTER system through the Metamask, which allows SC sign interactions (e.g., system login, AI services session agreement, AI session end request and other functionalities). In all pricing schemes, the agreement is reached when all the involved (mandatory) parties sign the SC(s).

Blockchain components consist of a public blockchain ledger, in our case Ethereum. Due to the high transaction costs and occasionally network congestion of the network in this article, we performed the evaluation on Binance Smart Chain (BSC) a blockchain network running SC-based applications implementing an Ethereum virtual machine (EVM). In other words, the proposed SCs can be deployed on both networks, which uses the same access credentials (mnemonic phrase and private keys). The developed SC templates can be used to support a variety of pricing schemes, see Section 6 for details. The distributed node infrastructure is maintained by the Ethereum and Binance community and offers a high level of distribution and availability. The main API technologies are written in the Java programming language. Hence, the core bridge to the Ethereum ledger is the Java library ethereumj[43] embedded in the core-layer API. Only those stakeholders that are registered by the SC's creator can trigger the deployed SC instances, that is, the DECENTER platform. Triggering executions and updates of the phases for all SCs is based on the actual content of the Ethereum blockchain network.

Furthermore, the backend that supports the blockchain event listeners (e.g., confirmed function executions of SC instances), is a Java-based Jersey RESTful microservice that manages the application workflow and acts as a bridge for the rest of the components. The video-streaming data is constantly generated by the surveillance camera and it is processed in close proximity to the camera. The context of the video stream is forwarded to the message broker, which is responsible for sending the data to the backend. The backend processes the input data and in case a person is detected on the video, it initiates the deployment of another container that has the AI method and model that will allow the detection of safety equipment. In order to do so, the backend: (1) retrieves the necessary container from the AI model repository; (2) triggers a SC execution on the blockchain; (3) once

given access to the AI container, it estimates an optimal infrastructure from the computing infrastructures pool that should host the container; (4) instructs the orchestrator (e.g., Kubernetes) on which node to deploy the new AI container; (5) instructs the orchestrator to stop the first container, because it is not needed anymore at that time. In the case of a safety violation, the backend is also responsible for generating a notification and sending it to the construction-site management team.

5.2.3 | Application capabilities

The smart application is designed to have the following capabilities:

- Detect and alert regarding dangerous situations during construction,
- Fog objectives: edge to cloud/edge offloading and the use of SCs for resources management,
- Digital representation: such as the position of the specific workers on the floor plan,
- AI objectives: distinguish objects, and use various AI models such as for object recognition, people identification, and prediction.

Once the very basic application has been developed, it can be gradually extended to incorporate different AI methods and models. The backend part is capable of integrating the information and based on already identified elements on the video-stream(s) to launch new AI-based microservices for further information processing. For this reason the orchestration process relies largely on the actual information required to address the specific scenario.

6 | SMART CONTRACTS DESIGN AND IMPLEMENTATION

Following the outline of the proposed architecture, our next goal was to study different pricing schemes that can be used by the stakeholders, end users and/or platform components (e.g., AI services) that contribute software services. We present a basic comparison between conventional and blockchain and SCs-based pricing schemes on the Ethereum/BSC network.

To start, the key properties of the Ethereum network related to monetization were aligned with three popular FIAT payment systems: Visa, MasterCard, and PayPal in order to analyze the feasibility. The key differences between FIAT and blockchain pricing schemes are:

- **Transaction processing fee** in cases of FIAT payment systems is based on the processed amount, approximately from 1.43% to 4.4%. On the other hand, the transaction fee in the case of blockchain ledgers is based on the complexity of the transaction (e.g., SC deployment, SC function trigger, etc.) and not on the amount of the coin/token to be transferred.
- **Operational service cost** is periodic in cases of FIAT payment systems. In cases of blockchain it is free in cases that the platform or merchant does not run its own node instance.
- **Advanced pricing schemes** is very limited in FIAT payment systems compared to the DeFi approaches. Additionally, SC custom-defined functions allow a higher level of transaction flexibility. For example, lock-in transaction fund refers to the actual locking of funds in the SC until certain functional conditions have been reached to release the funds, or just fractioned funds, to the appropriate user entity. In alternative cases, these mechanisms are executed off-chain, without direct access by the end user by trusted third party services (such as trusted smart oracles).

6.1 | Pricing schemes

Following a detailed analysis of the various stakeholder needs, a variety of pricing schemes were identified for implementation using SCs. They are based on actual usage patterns of cloud-computing resources and services. A specific problem that is addressed is the income division among multiple parties, including cloud providers, CI deployment platforms, software service providers, monitoring services, and similar.

Figure 4 is drawn to highlight the differences between the various SCs that offer various degrees of flexibility. The AI-based service process flow in the context blockchain-as-a-service (BaaS) is explained for the various SCs in the following.

Fixed price is a basic monetization use case. The DECENTER platform offers the usage of the AI session on demand for a price and period, both fixed. After the DECENTER platform deploys the SC instance, the primary end user requests the AI session by triggering the respective function and thus sending the ETH funds. Upon successful payment, the event notifies the DECENTER platform and initializes the docker container instances as a AI session.

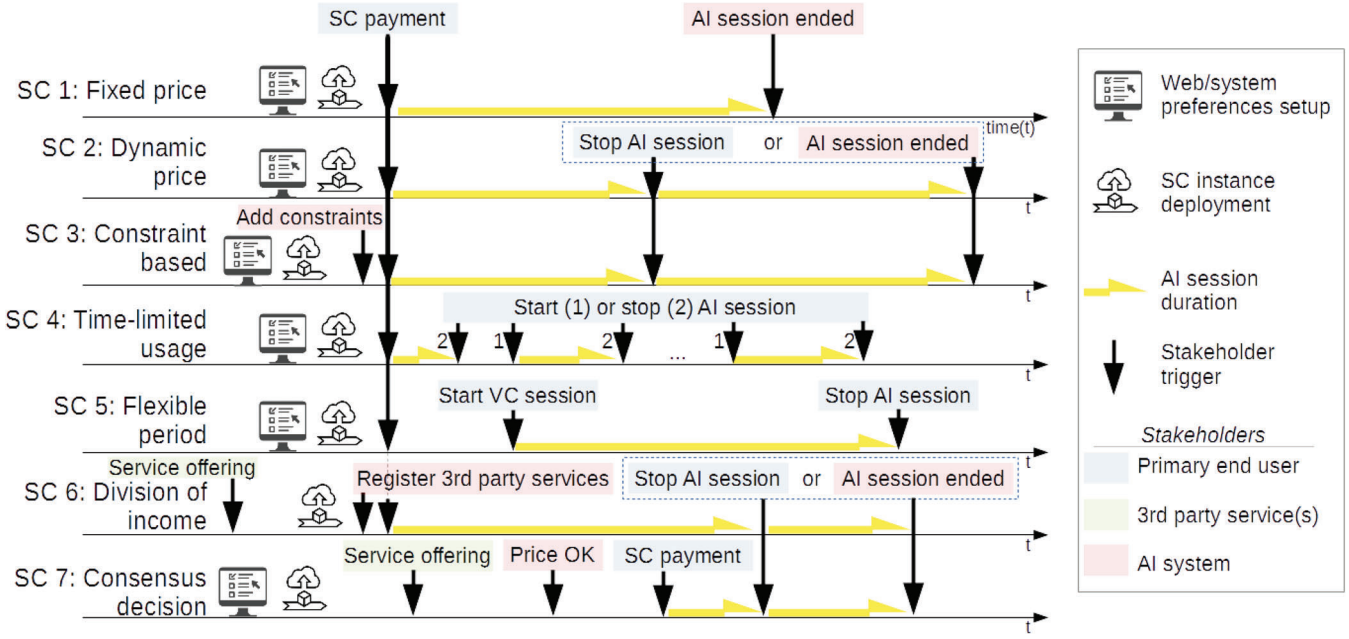

FIGURE 4 Differences in the processes of using the various SCs

Dynamic price complements the fixed-price use case and involves the same stakeholders (AI session and primary end user). Its dynamic pricing offers higher flexibility based on the actual AI session availability. For example, it can be used when the end user knows the maximum time needed for a video session. After an agreement is reached, the user pays the full price. The ETH funds are locked by the SC. The funds may be unlocked to allow for extra payment, if the maximum period is reached and the video session is continued, or if the end user stops using the AI session by triggering the SC stop function. In the unlocking phase, the actual usage time is charged—the proportional ETH funds of unused time is returned to the end user, while the rest is sent to the AI session.

Time-limited usage refers to the use case when the primary end user agrees in advance the per-minute conversation price, buys a specific number of AI session minutes in advance and uses them gradually for AI sessions on demand. In this use case, the DECENTER platform deploys and initializes the SC, while the end user triggers the AI session start/stop actions through the SC function linked to an event listener in our DECENTER platform. A typical usage scenario is a Big-Brother-like TV show and for other content provided in real-time online.

Flexible usage period offers more flexibility for the end users that cannot define the exact period when the AI session will be used. In the proposed end user's period, the AI session has to be either available or the deployment of the AI session has to be optimized, matching a deployment for high QoS, which is more effective. The charging methodology follows the dynamic-price use case. It includes stakeholders who facilitate the high QoS of the service, for example, by providing mechanisms for the fast deployment of the containerized AI session for a specified time period. The SC allows these stakeholders to place charges for the provisioning of their specific services.

Division of income is a particular monetization process where any additional parties, besides the AI session provider and the primary end user, participate in the provisioning of the AI session, that is, its software or hardware infrastructure or provided content. For example, one provider offers specific AI containers, another (cloud provider) offers infrastructure and a third party offers other services, such as QoS monitoring. The division of income is agreed upon by the parties in advance and is written at the instance of the SC, for example, the price per hour for using the specific service. An overall advantage of this approach for the primary end user is reflected as a lower overall price of the AI session. In addition, this approach aims to collaboratively include specific services, in the cloud or even in the fog, that do not need to be maintained in the infrastructure of the DECENTER platform owner.

Consensus decision is an upgrade of the division-of-income use case. The consensus is reached through a democratic voting SC, similar to the one proposed by Bragagnolo et al.[44] As a result, the management of the AI session is divided among parties specialized for different infrastructure or service aspects and thus, the AI session benefits from improved QoS, availability and overall cost reduction. The primary end user is not directly aware of the ETH funds being distributed among the AI-enabling services, although this fact is evident from the SC instance. The distribution of ETH funds is always performed after the usage of the AI session, when all the costs in terms of Ethers are distributed accordingly to the consensus rules to individual involved roles.

Constraint based focuses on the legal aspects determined by the end user's constraints. As an example, the European Union's General Data Protection Regulation (GDPR) compliance can be agreed by all the end users through the SC. In some cases, a geographical definition is also essential

and should be determined by the primary end user and the DECENTER platform in advance (e.g., not all the data services are allowed to move outside the European Union for data protection). In general, by constraint limiting, the price for the end user can change (increase or decrease) based on the required high or not-so-high QoS, such as video-frames resolution or frames-per-second rate.

6.2 | Walk-through for consensus decision smart contract

In this section we systematically describe the workflow of the consensus-decision pricing scheme SC. This use case is motivated by the needs of transparency in the definition of agreements among the primary end user, the DECENTER platform and any third-party services through an Ethereum SC.

The aim of the consensus-decision pricing scheme is primarily the possibility to involve the DECENTER platform and a third-party service to determine the AI session price for the primary end user. Another important aspect follows the dynamic-price-scheme approach properties that are summarized as increased flexibility for the primary end user, who cannot estimate the exact duration for the required AI session. In comparison to traditional investigated pricing-scheme approaches (such as PayPal, MasterCard, and Visa), the SC approach is transparent, as it is not based on a centralized authority, and uses the defined SCs' functions in the solidity programming language.

The entities involved in the pricing-scheme process are: (i) the primary end user who initiates the AI session, (ii) the blockchain component that provides the SC execution, (iii) the solution services including third-party entities, and (iv) the cloud providers that offer computing resources for the AI application components as third-party services.

The flow of interactions between the blockchain component and the triggering entities is depicted in Figure 5. In the beginning, the primary end user sets basic properties through the web GUI, for example, the pricing-scheme type (the consensus-decision pricing scheme SC7), the maximum time period (one hour) and determines additional QoS preferences (e.g., preferred video quality, approximate number of participants). Based on the primary end user's properties, the blockchain service deploys (function *deploy()*) an SC instance. In the next stage third-party services register their pricing policies (function *determineStakeholdersPrice()*) and on success they receive an event-based trigger *StakeholderPriceNotification()*. The DECENTER platform confirms the consensus through the function *consensusFinished()* and sends the notification about the price of the AI session to the web GUI. The approval for the SC pricing scheme is declared when the primary end user pays the full incentive (function *payVCService()*) that triggers the AI session deployment on a cloud provider. After a successful AI session deployment the primary end user receives a unique uniform resource locator (URL) that can be shared among the other AI service participants. Although it is common practice to share unique public URLs, this can be additionally enhanced, for example, by registering individual participants' addresses to obtain the AI session access. The AI session ends automatically when the duration determined in the SC is exceeded or on demand by the primary end user (function *triggerReleaseTime()*) where the return of the ETH funds to the primary end user needs to be calculated. In both cases the AI session un-deployment process is triggered and the AI CI un-deploys and thus frees the allocated resources. Finally, the primary end user receives the notification about the successful completion of the SC.

To outline the ETH funds locking functionality it is mandatory to define some key global attributes—lock time in seconds determined by a current block timestamp, session start time, maximum price (referred to lock time in seconds), end-user address and AI owner address. Due to the use of block-based generation of the timestamp, the overall delay is inducted between the discrete time-generation of blocks, which is on average 15 s[45] and also by the average deployment time of the AI session that is mostly affected by the cloud provider as described in the following section.

An important aspect in SC-functions management is the event triggers. In our case, after the end user triggers the payment function our DECENTER platform needs to be notified as an event. An option to do that is to either develop the service with an Ethereum-based block listener (e.g., replay filters in the web3j library) or use the Ethereum's alarm clock service,[46] which is designed to serve SC events.

A fundamental SC template that locks the ETH funds of the sender is described in Algorithm 1.

Algorithm 1: Example SC function payVCService() for time-based locking of Ethereum funds in dynamic pricing scheme use case.

```
input data: object payable{addressFrom, value}
return: Boolean success

// in case the ETH value does not match the agreed price
IF check basic conditions {
   emit event(success = false, 0, 0)

} ELSE {
   // set the release time, address of the sender and deposit ETHs,
   // while the lockTimeSeconds is a global attribute
   releaseTime = currentBlockTime + lockTimeSeconds;
```

```
    balance = balance + value;
    globalAddressFrom = addressFrom;
    isFinished = false;
    emit event(success = true, currentBlockTime, releaseTime);
}
```

Another pillar function is the AI session-stopping (termination, un-deployment) notification that is triggered by the primary end user. After the successful trigger, the ETH transaction to the AI session address is performed and another one to the address of the primary end user according to the duration of the VC service.

6.3 | Implementation details

The development process of SCs started with an environment setup, where we aimed to facilitate the integration of existing components by using the Java programming language in the back-end entirely, while in the front-end we used the standard JavaScript approach that enables the integration with the MetaMask bridge.

The development of SCs proceeded as follows: (i) For each pricing-scheme use case we prepared only one individual SC to reduce the cost of the deployment of another SC and consequently the delay of the deployment operation. A disadvantage of this approach is the potentially decreased scalability, which may be required when using multi-party SCs initializing new ones; (ii) The SCs specific for the AI session needed to be carefully designed and validated (e.g., by using the Oyente SC validation tool), analyzed and evaluated at different levels, such as code pattern comparison and simulation of actual use with multiple parties to the same SC. Besides that, the OpenZeppelin framework provided reusable tested SCs templates for Ethereum; (iii) The initial state of all deployment addresses owned by the DECENTER platform contained 0 ETH; (iv) The implementation followed the object-oriented programming approach, especially the inheritance of SCs, as shown in Figure 6. The actual solidity code of all the pricing-scheme use cases is available in the BitBucket repository;[47] and (v) Actual behavior of SCs had to be observed (e.g., time elapsing of individual functions, cost estimation of operations and so on) in a main or testing Ethereum environment network as presented in Section 7.

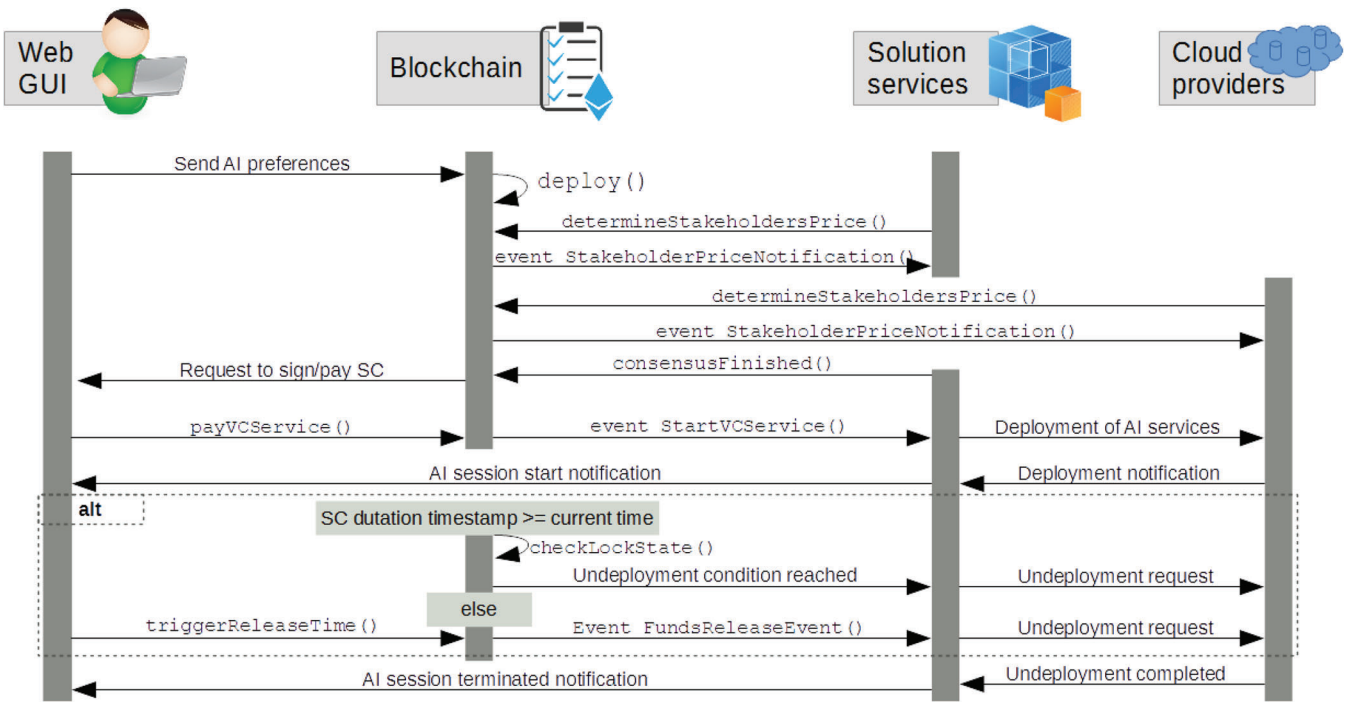

FIGURE 5 Sequence diagram of the consensus decision pricing scheme use case (SC7)

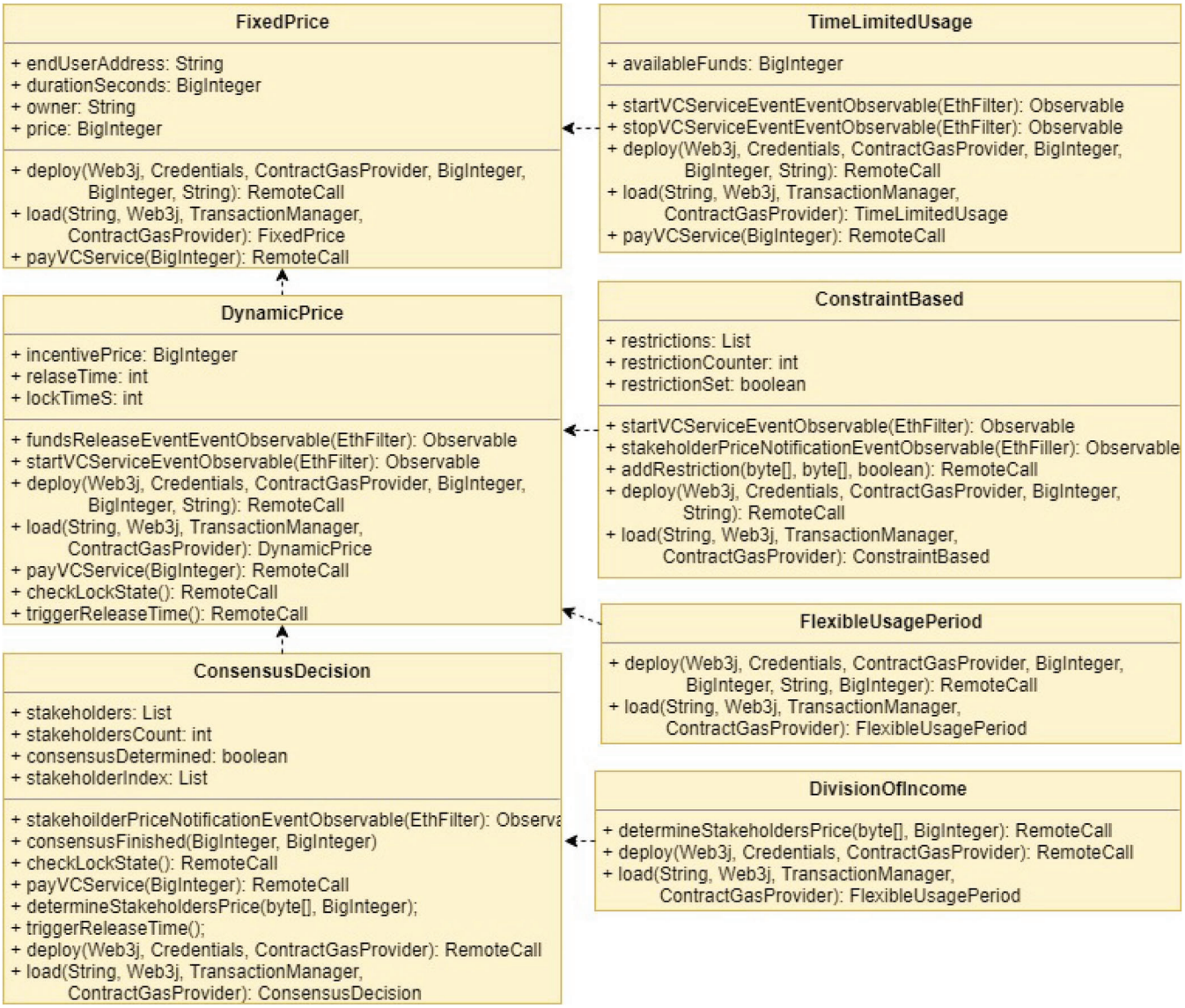

FIGURE 6 The class diagram represents SCs' main public attributes, methods, and inheritance class relationships

7 | EXPERIMENTAL STUDY

The SCs were implemented on the Ethereum testnet network, for which we performed various experiments. The experimental methodology and results are elaborated in this section.

7.1 | Experimental methodology

The walk-through example of the dynamic pricing scheme SC presented in the previous section gives us an idea of the possible interactions among the service components. Here, we analyze the designed architecture and SCs by collecting various metrics, such as the *execution time* and the *transaction cost* of the SCs for the involved stakeholders. We conducted a series of DECENTER platform runs using the implemented SC use cases on the Rinkeby Ethereum testnet environment using PoA[48] transaction and the more affordable BSC testnet.

The experimentation workflow is shown in Figure 7. In the SCs development phase we use a variety of tools (validate, optimize, design pattern frameworks and others) that are described in Section 2. In the next phase, the experimental setup consists of generating testing addresses for all stakeholders, except for the enabling services that are multiple for each role (e.g., monitoring system, orchestrator, and cloud provider). In order to execute transactions, all the addresses need to contain ETH funds that are claimed through the so-called Authenticated Faucet.

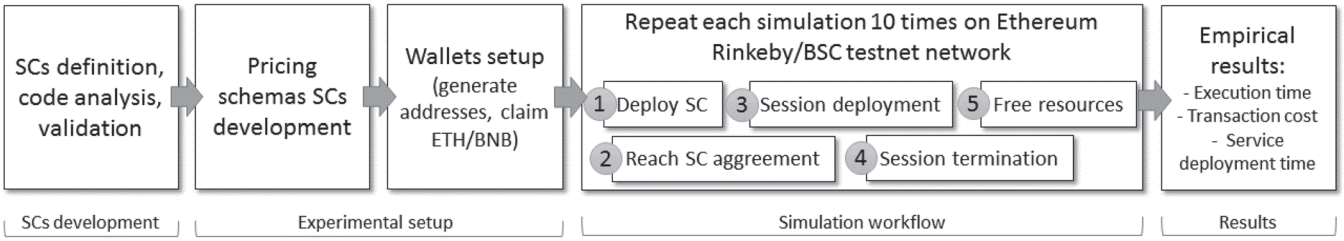

FIGURE 7 Experimentation workflow for the implemented architecture and smart contracts (SCs)

The DECENTER platform life-cycle is represented with the following steps:

1. Deployment of the SC on an address accessible from a running DECENTER platform.
2. SC agreement is reached through function triggering depending on the monetization use case.
3. An actual AI session is deployed according to the SC configuration specified by the end-user (e.g., geolocation, required SLOs).
4. The AI session is terminated either when the reserved time is exceeded or on demand by the primary end user of the AI session, and thus the SC event notifies the DECENTER platform.
5. The DECENTER platform resources are released.

This workflow allows us to obtain Ethereum-testnet-based quantitative metrics such as the execution time and the transaction cost. These metrics can be further objectively mapped to Ethereum mainnet environment and are discussed in the following sub-section. Furthermore, we measured the deployment time of our VC CI at cloud providers located at five different geolocations, which is an indication of a collected QoS metric.

7.2 | Results

The experimental environment is based on running our own testnet Ethereum node (Rinkeby) to fully exploit the network performance. For example, by using the common alternative Infura Web tool[49] with the API-based access to the Ethereum testnet node we are not able to run observers for blocks and confirmed or pending transactions. Since we run our own Ethereum testnet node, we can observe the updates of the ledger after the execution of the (synchronous) transaction and at the same time observe (asynchronously) the content of the new blocks added to the blockchain, and thus determine the end time of the transaction based on which triggers first. To integrate the SCs with other services we used the Ethereum virtual machine (EVM) events. All the available monetization use cases (SCs) were run 10 times, and the average results are presented.

Figure 8 shows that the execution time of the transactions triggered by the stakeholders is mainly affected by their quantity, where a theoretical minimum for each transaction is the block generation time, which is approximately 15 s. Furthermore, the deployment of SCs generally takes the time of two blocks cycle generation to be included in the block. In the case of time-limited usage, the execution time of the end user may be affected drastically; however, it mainly consists of triggering starting/stopping the AI session 5 times. In the stopping phase, it does not affect the QoS provided to the end user. In the division of income and consensus decision use cases, the enabling services interact with the SCs; however, these transactions also do not affect the provided QoS to the end users (i.e., in our illustrative case, the AI-service launching time).

Figure 9 describes the cost of the transactions using the default GAS provider. On average, the DECENTER platform transactions are the most expensive due to the larger data size that needs to be written in the blocks (e.g., the deployment of SCs). Additionally, the transaction cost is affected by the selection of input data and the quantity of the function attributes. In the case of the constraint-based use case, the constraints, defined as a table of bytes (*byte32*), are significantly higher than the numerically defined values, such as *integers*. In cases when multiple different stakeholders are interacting with the SCs (i.e., the division of income and consensus decision), the transaction costs are 8 times lower compared to the other use cases.

After each successful monetization agreement is reached among the primary end user and the AI session, the actual deployment time of the DECENTER platform is measured. This reflects an important aspect of the QoS of the provided AI service. The duration of the orchestration process is a critical parameter, which is needed to calculate the actual running time of the AI session because it needs to be summed up to the time needed to write the SC agreement into the block. Our Kubernetes orchestrator runs the service across the following infrastructure: 4 vCPUs with hardware-accelerated CPU virtualization powered by Intel Xeon E5649 CPUs clocked at 2.53 GHz, 2 GB of RAM and 20 GB of HDD. In the deployment experiment, we deployed the CI AI session 10 times at different geolocations, as presented in Table 2.

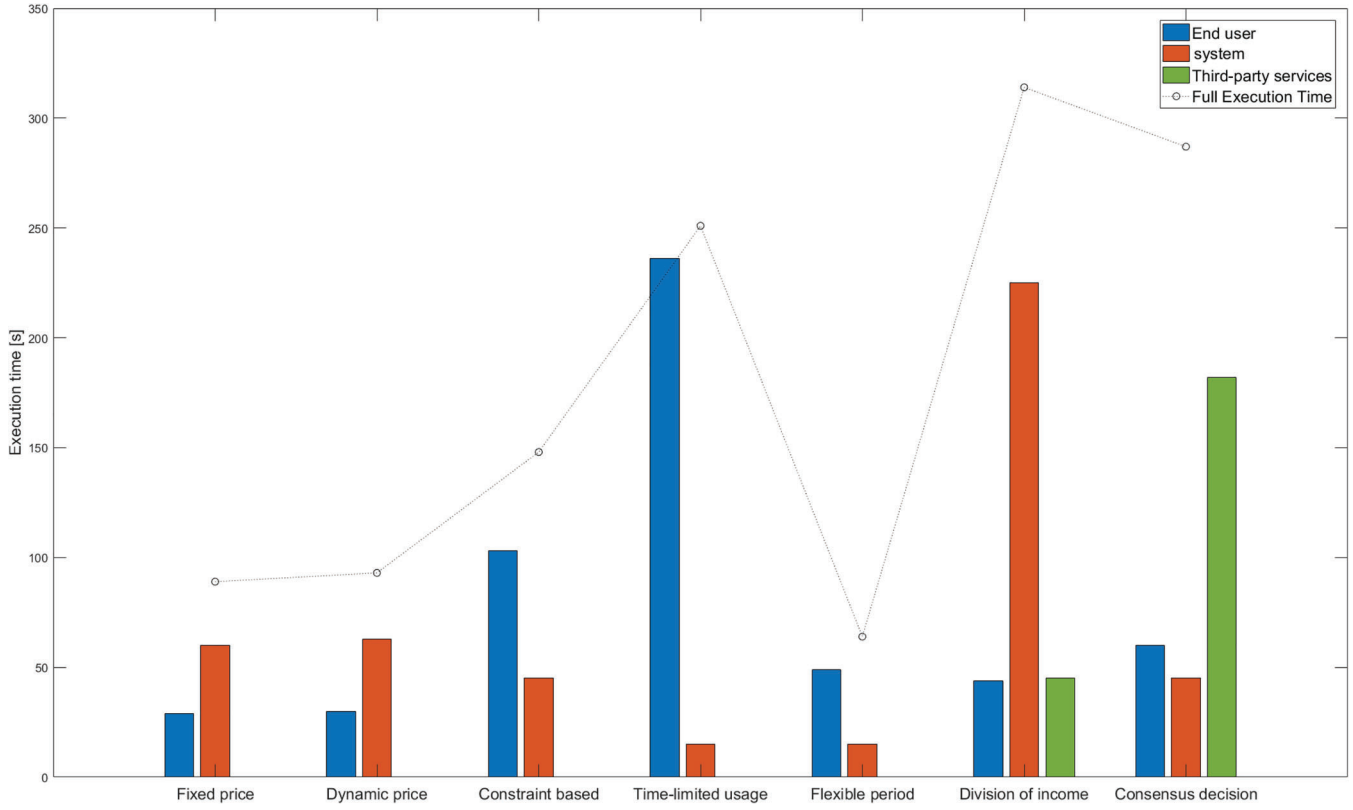

FIGURE 8 Performance results of SCs among different stakeholders on the Ethereum Rinkeby testnet environment

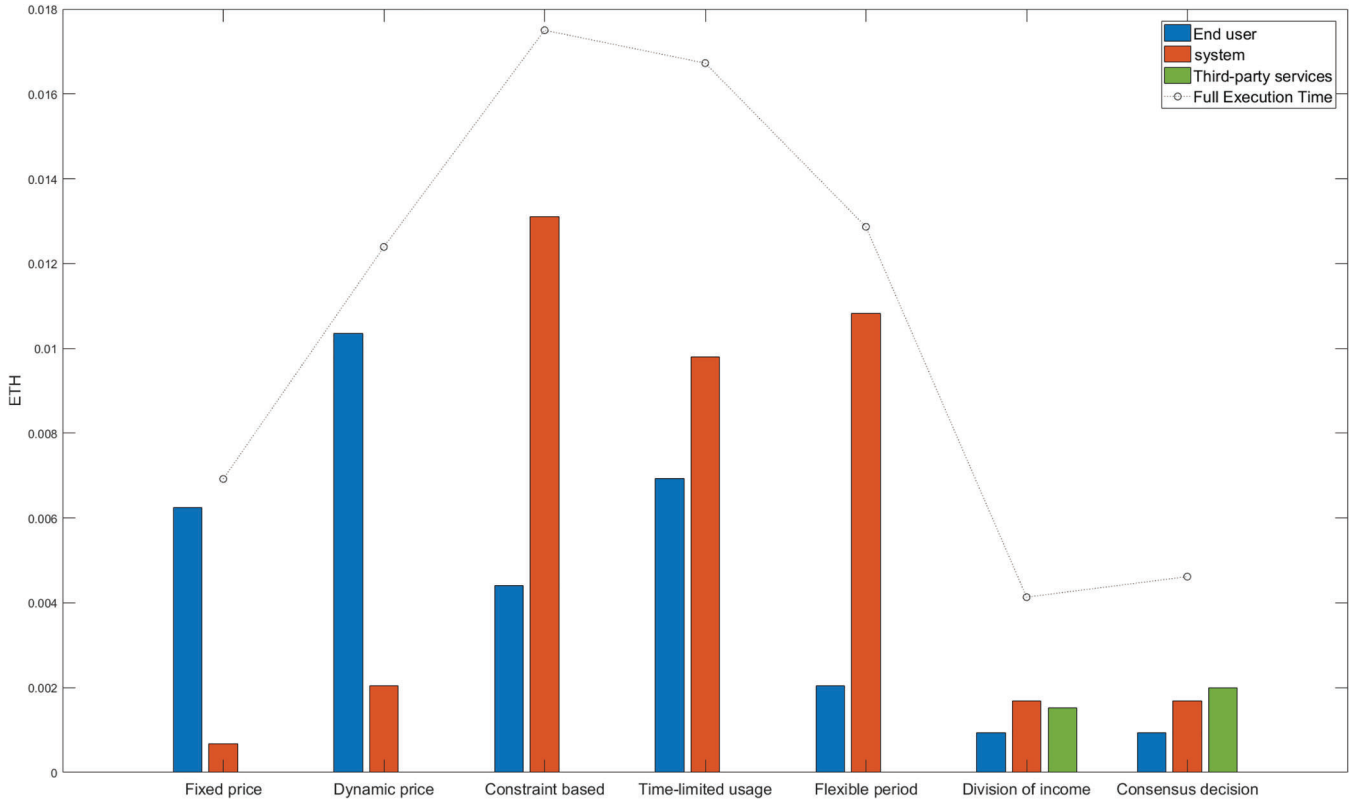

FIGURE 9 Comparison of GAS cost estimation when executing multi-party SCs

TABLE 2 Deployment times of CI based AI session on different geolocations

Cloud name	Continent	Deployment time (ms)
Arnes	Europe	3133
flexiOps	UK	8033
GKE-EU-WEST	Europe	1866
GKE-ASIA-EAST	Asia	5133
GKE-US-WEST	USA	2266

8 | DISCUSSION

Some of our initial use-case scenarios required fast response times for time-critical operations, for example, when detecting whether a person on the construction site, who is observed by a camera, wears a hard hat or not. Fast response times are guaranteed by DECENTER's QoS algorithms that are implemented along with a Kubernetes controller that employs an advanced MDP algorithm for cloud options ranking, formal assurances and verification. The ability to quickly switch between the AI methods that operate on top of a specific video stream, for example, after detecting that the person does not wear a helmet, two additional methods are started that should detect whether the person wears a safety vest and the color of the vest. This is achieved by the DECENTER's technology for a trustless, reliable, transparent, flexible and even consensus- enabling brokerage of computing resources and orchestration through the proposed SCs that cover different typologies of interactive entities. These services need to be deployed on demand at different geolocations to follow network-based and other requirements. Our service providers indicate that the services may be deployed in a reasonable time for our domain, with a mean value in 4086 ms and a median value of 3133 ms.

Furthermore, by using SCs a multi-party SLA can be set up more transparently on heterogeneous cloud architectures. The results show that in practice it is better to have fewer functions in the SCs so as to reduce the transaction times, but on the other hand, the input data of SCs should be pragmatically defined by using minimally required sizes of the data types and thus lowering the transaction costs.

In the experimental study, we analyzed individual pricing schemes with the intent to prove the feasibility of the proposed architecture. With the introduction of a blockchain layer, the system is implemented as a distributed Ethereum node storage, and it represents a full environment that uses SCs. The running node requires upwards of 120 GB of storage and 8 GB+ of memory. For each monetization use case, we can determined the transaction costs for each stakeholder involved, which is significantly lower than $ 1 per transaction while using a BSC network. The SCs can be additionally improved in the case that the SCs would be ported to alternative node layers such as side chains. In such case the solution may also be applied to the domain of IoT, which has requirements for micro transactions and other related domains.

9 | CONCLUSION

In this article we presented an architecture for the monetization of a DECENTER platform that follows the pay-as-you-go BaaS concept. Our approach was demonstrated using seven SCs, including ones that support multi-party stakeholders, consensus, governance and block-time triggered events. Compared to the traditional *monetization out* system, this new architectural approach can offer significantly lower costs (e.g., transaction, operational and other), even more so if using a BSC instead of an Ethereum network. Additionally, a trustless layer can be implemented by using SCs and records of the provided QoS as evidenced by the provided example. Our study supports the conclusion that SCs are suitable for establishing transparent mutual agreements among end users and AI services (such as video-conferencing and other video-based services) that may be expected in the near future. This new architecture, therefore, facilitates the overall payment process without any additional third-party costs for the service owners (such as banks, payment cards and other transaction fees).

Currently, the central paradigm with SCs is the management of the data received from outside a blockchain from the so-called smart oracles. Smart oracles are still in their infancy. However, they hold the potential to provide the trusted data needed for the execution of SCs. As an example, a smart-oracle-based monitoring system can provide independent QoS metrics used to renegotiate the price in cases when our DECENTER platform would not be sufficient to enforce an SLA based on user stated SLOs. We intend to exploit the potential of SCs systematically to deliver a blockchain recommender system in our future work. The purpose of the blockchain recommender system would be to facilitate the SCs' integration in cloud-computing architectures. Moreover, the research study proposes the use of smart oracles as mechanisms for dealing with off-chain data that can further contribute to the seamless integration of applications and their deployment in cloud-computing architectures. With such an approach it would be possible to enable the integration of potential SCs in cloud, fog, and edge architecture.

ACKNOWLEDGMENT

The DECENTER project has received funding from the European Union's Horizon 2020 research and innovation programme under Grant agreement No. 815141.

DATA AVAILABILITY STATEMENT

Prototype Multi-Party SLA Smart Contracts at https://bitbucket.org/sgec/prototype-multi-party-sla-smart-contracts/src/master/, Reference number 48.

ORCID

Sandi Gec https://orcid.org/0000-0003-2331-4523
Vlado Stankovski https://orcid.org/0000-0001-9547-787X

REFERENCES

1. DECENTER project. Accessed October 12, 2021 https://www.decenter-project.eu/
2. Kochovski P, Stankovski V. Supporting smart construction with dependable edge computing infrastructures and applications. *Automat Construct*. 2018;85:182-192. doi:10.1016/j.autcon.2017.10.008
3. Štefanič M, Stankovski V. A review of technologies and applications for smart construction. *Proc Inst Civil Eng Civil Eng*. 2019;172(2):83-87. doi:10.1680/jcien.17.00050
4. Anjum A, Sporny M, Sill A. Blockchain standards for compliance and trust. *IEEE Cloud Comput*. 2017;4(4):84-90. doi:10.1109/MCC.2017.3791019
5. Ethereum (Blockchain application platform). Accessed December 18, 2020. https://www.ethereum.org/
6. IOTA. Accessed December 18, 2020. https://iota.org/
7. Ripple. Accessed December 18, 2020 https://ripple.com/
8. Stor J. Accessed December 18, 2020 https://storj.io/
9. Haber S, Stornetta WS. How to time-stamp a digital document. *J Cryptol*. 1991;3(2):99-111. doi:10.1007/BF00196791
10. Nakamoto S. Bitcoin: a peer-to-peer electronic cash system. http://bitcoin.org/bitcoin.pdf
11. Buterin V. Ethereum white paper, updated September 30, 2015. Accessed October 30, 2015. https://github.com/ethereum/wiki/wiki/White-Paper.
12. Sillaber C, Waltl B. Life cycle of smart contracts in blockchain ecosystems. *Datenschutz und Datensicherheit - DuD*. 2017;41(8):497-500. doi:10.1007/s11623-017-0819-7
13. Wöhrer M, Zdun U. Design patterns for smart contracts in the ethereum ecosystem. Proceedings of the 2018 IEEE International Conference on Internet of Things (iThings) and IEEE Green Computing and Communications (GreenCom) and IEEE Cyber, Physical and Social Computing (CPSCom) and IEEE Smart Data (SmartData); 2018:1513-1520.
14. Truffle suite. Accessed December 18, 2018. https://truffleframework.com/
15. Zhang F, Cecchetti E, Croman K, Juels A & Shi E Town Crier: an authenticated data feed for smart contracts cryptology eprint archive. Report 2016/168; 2016. https://eprint.iacr.org/2016/168
16. Kosba A, Miller A, Shi E, Wen Z, Papamanthou C. Hawk: the blockchain model of cryptography and privacy-preserving smart contracts. Proceedings of the 2016 IEEE Symposium on Security and Privacy (SP); 2016:839-858.
17. Luu L, Chu D-H, Olickel H, Saxena P, Hobor A. Making smart contracts smarter. In: Edgar W, ed. *CCS '16*. ACM; 2016:254-269.
18. Chen T, Li Z, Zhou H, et al. Towards saving money in using smart contracts. In: Andrea Z, Sven A, eds. *ICSE-NIER '18*. ACM; 2018:81-84.
19. TetriX. December 18, 2018. https://github.com/Decentrix/TetriX
20. OpenZeppelin. December 18, 2018. https://github.com/OpenZeppelin/openzeppelin-solidity
21. Swan M. Anticipating the economic benefits of blockchain. *Tech Innov Manag Rev*. 2017;7(10):6-13. doi:10.22215/timreview/1109
22. Yoo S. Blockchain based financial case analysis and its implications. *Asia Pacific J Innovat Entrepreneurship*. 2017;11(3):312-321. doi:10.1108/APJIE-12-2017-036
23. Ankenbrand T, Denis B. A structure for evaluating the potential of blockchain use cases in finance. *Perspect Innovat Econom Bus*. 2018;17(2):77-94. doi:10.15208/pieb.2017.06
24. Peter H, Moser A. Blockchain-applications in banking & payment transactions: results of a survey. *Eur Financ Syst*. 2017;141:141-149.
25. Paščinski U, Trnkoczy J, Stankovski V, Cigale M, Gec S. QoS-aware orchestration of network intensive software utilities within software defined data centres. *J Grid Comput*. 2018;16(1):85-112. doi:10.1007/s10723-017-9415-1
26. Efthymiopoulou M, Efthymiopoulos N, Deltouzos K, Christakidis A. Robust control in cloud assisted peer to peer live streaming systems. *Pervas Mob Comput*. 2017;42:426-443. doi:10.1016/j.pmcj.2017.06.005
27. Al-Turjman F. Price-based data delivery framework for dynamic and pervasive IoT. *Pervas Mob Comput*. 2017;42:299-316. doi:10.1016/j.pmcj.2017.05.001
28. Hu B, Zhang Z, Liu J, et al. A comprehensive survey on smart contract construction and execution: paradigms, tools, and systems. *Patterns*. 2021;2(2):100179. doi:10.1016/j.patter.2020.100179
29. Lu Q, Xu X. Adaptable blockchain-based systems: a case study for product traceability. *IEEE Softw*. 2017;34(6):21-27. doi:10.1109/MS.2017.4121227
30. Xia Q, Sifah EB, Asamoah KO, Gao J, Du X, Guizani M. MeDShare: trust-less medical data sharing among cloud service providers via blockchain. *IEEE Access*. 2017;5:14757-14767. doi:10.1109/ACCESS.2017.2730843
31. Kochovski P, Stankovski V. Building applications for smart and safe construction with the DECENTER fog computing and brokerage platform. *Automat Construct*. 2021;124:103562. doi:10.1016/j.autcon.2021.103562
32. Gec S, Paščinski U, Stankovski V. Semantics for the cloud: the ENTICE integrated environment and opportunities with smart contracts. Time Critical Applications and Infrastructure Optimization the 3rd Edition in the Series of Workshop on Interoperable Infrastructures for Interdisciplinary Big Data Sciences (IT4RIs 18); 2018. 10.5281/zenodo.1163999
33. Venus decentralised marketplace. Accessed March 27, 2021 https://venus.io/

34. Gürgen L, Munilla C, Druilhe R, Gandrille E, Nascimento J, eds. *sensiNact IoT Platform as a Service*. Vol 6. John Wiley & Sons, Ltd; 2016:127-147.
35. Eclipse SensiNact. Accessed January 18, 2021. https://projects.eclipse.org/proposals/eclipse-sensinact
36. Bonomi F, Milito R, Zhu J, Addepalli S. Fog computing and its role in the Internet of Things. In: Mario G, Dijiang H, eds. *MCC '12*. Association for Computing Machinery; 2012:13-16.
37. Binz T, Breitenbücher U, Haupt F, et al. OpenTOSCA – A runtime for TOSCA-based cloud applications. In: Samik B, Cesare P, Liang Z, Xiang F, eds. *Service-Oriented Computing*. Springer; 2013:692-695.
38. Kochovski P, Drobintsev PD, Stankovski V. Formal quality of service assurances, ranking and verification of cloud deployment options with a probabilistic model checking method. *Inf Softw Technol*. 2019;109:14-25. doi:10.1016/j.infsof.2019.01.003
39. Faticanti F, De Pellegrini F, Siracusa D, Santoro D, Cretti S. Cutting throughput with the edge: app-aware placement in fog computing. Proceedings of the 6th IEEE International Conference on Cyber Security and Cloud Computing, CSCloud 2019 / 5th IEEE International Conference on Edge Computing and Scalable Cloud, EdgeCom 2019, Paris, France, June 21-23, 2019:196-203; IEEE.
40. Gec S, Lavbič D, Bajec M, Stankovski V. Smart contracts for container based video conferencing services: architecture and implementation. In: Massimo C, Emanuele C, eds. *Economics of Grids, Clouds, Systems, and Services*. Springer International Publishing; 2019:219-233.
41. Abadi M, Barham P, Chen J, et al. TensorFlow: a system for large-scale machine learning. In: Kimberly K, Timothy R, eds. *OSDI '16*. USENIX Association; 2016:265-283.
42. MetaMask. December 18, 2018. https://metamask.io/
43. ethereumj. December 18, 2018. https://github.com/ethereum/ethereumj
44. Bragagnolo S, Rocha H, Denker M, Ducasse S. SmartInspect: solidity smart contract inspector. Proceedings of the 1st International Workshop on Blockchain Oriented Software Engineering IEEEIWBOSE 2018; Campobasso, Italy.
45. Ethereum statistics by Etherscan service. Accessed June 9, 2018. https://etherscan.io/chart/blocktime
46. Ethereum alarm clock service. Accessed June 9, 2018. http://www.ethereum-alarm-clock.com/
47. BitBucket repository containing SC sources. Accessed December 18, 2018. https://bitbucket.org/sgec/multi-party_service_level_agreements_with_smart_contracts/src
48. Ethereum testnet network Rinkeby. Accessed December 18, 2018. https://www.rinkeby.io
49. Infura web service. Accessed December 18, 2018 https://infura.io

How to cite this article: Gec S, Kochovski P, Lavbič D, Stankovski V. Multi-party smart contract for an AI services ecosystem: An application to smart construction. *Concurrency Computat Pract Exper*. 2022;e6895. doi: 10.1002/cpe.6895