IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received January 30, 2020, accepted February 13, 2020, date of publication February 18, 2020, date of current version February 27, 2020.

Digital Object Identifier 10.1109/ACCESS.2020.2974810

A Blockchain-Based Approach for the Creation of Digital Twins

HAYA R. HASAN[1], KHALED SALAH[1], RAJA JAYARAMAN[2], MOHAMMED OMAR[2], IBRAR YAQOOB[1], SAŠA PESIC[3], TODD TAYLOR[3], AND DRAGAN BOSCOVIC[3]

[1]Department of Electrical Engineering and Computer Science, Khalifa University, Abu Dhabi, UAE
[2]Department of Industrial and Systems Engineering, Khalifa University, Abu Dhabi, UAE
[3]ASU's Blockchain Research Laboratory, Arizona State University, Tempe, AZ 85281, USA

Corresponding author: Ibrar Yaqoob (ibraryaqoob@ieee.org)

This work was supported by the Khalifa University of Science and Technology under Award CIRA-2019-001 and RCII-2019-002-Research Center for Digital Supply Chain and Operations Management.

ABSTRACT The rapid advancements in computing, storage, communications, and networking technologies have enabled the creation of Digital Twins (DTs). A DT is a digital representation of a real-world physical component, product, or equipment. A DT can be used for 3-D design, testing, simulation, and prototyping prior to the manufacturing of the physical component. Once a physical component is in operation, a DT can be used for configuration, monitoring, diagnostics, and prognostics. It is expected that DTs will gain significant attention in the foreseeable future, and will play a key role in Industry 4.0. However, today's approaches, systems, and technologies leveraged for the creation of DTs are mostly centralized and fall short of providing trusted data provenance, audit, and traceability. Also, data related to transactions, logs, and history are not secure or tamper-proof. In this paper, we propose a blockchain-based creation process of DTs to guarantee secure and trusted traceability, accessibility, and immutability of transactions, logs, and data provenance. Our proposed approach uses smart contracts to govern and track transactions initiated by participants involved in the creation of DTs. Our approach also employs decentralized storage of interplanetary file systems to store and share DTs data. Moreover, we present details on our system design and architecture, implementation, and algorithms. Furthermore, we provide security and cost analysis, and show how our approach fulfills the requirements of DTs process creation. We make the smart contract code for creating DTs publicly available on Github.

INDEX TERMS Digital twins, blockchain, Ethereum, smart contracts, security, Industry 4.0.

I. INTRODUCTION

With unprecedented proliferation of Cyber-Physical Systems (CPS), advanced data analytics, and high performance computing technologies, Industry 4.0 has brought extensive changes to industrial operations since the past decade. A Digital Twin (DT) is considered as an indispensable component of the CPS concept which refers to a virtual representation of any physical object, process, system, smart city, to name a few [1]. DTs aim to optimize industrial operations and maintain physical assets and manufacturing processes before their creation. An amalgamation of physical and virtual digitalized elements leads to the creation of DTs. These DTs operate in the digital world just like their physical twins in the real physical world [2]. Figure 1 shows a digital replica of a physical engine in a three-dimensional model. DTs bridge the gap between the real and virtual worlds. The pairing between the digital and physical worlds using streams of data generated by sensory devices not only helps in terms of proactive maintenance but also helps in building predictive simulation models [3]. Moreover, performance optimization is one of the prime aspects in industries that can be achieved by creating DTs of real devices including aircraft engines, smart containers, wind turbines, space crafts, and others. Furthermore, DTs have great potential to improve the healthcare industry. For example, a healthcare provider can simulate an operation on a patient's virtual organ and perform experiments on it with different procedures before performing the real operation on the patient's physical organ [4].

The associate editor coordinating the review of this manuscript and approving it for publication was Vicente Alarcon-Aquino.

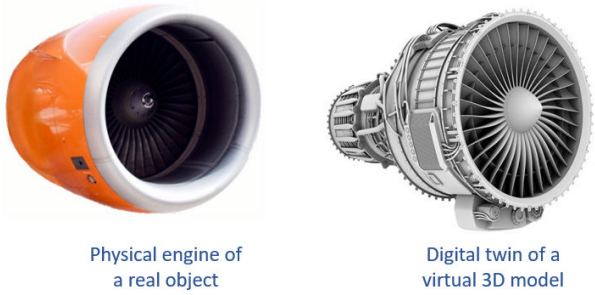


FIGURE 1. A real physical engine vs. its digital twin.

The concept of DTs was introduced by NASA [5], [6] to find effective ways to resolve challenges associated with the Apollo 12 mission. Specifically, the organization needs to operate on such systems that are out of their physical proximity and thereby led toward the creation of a ''virtual model of a process or service'' that has similar characteristics to the physical object [7]. The augmentation and integration of the virtual and real worlds are established using modeling techniques, Internet of things, artificial intelligence, and data analytics of the uninterrupted data streams collected from sensory elements. It is anticipated that half of the large industrial companies will adopt DTs by 2021, and their market is set to hit $15.66 billion by 2023 [7]. Hence, DTs will become prevalent in smart industries.

DTs can mainly be classified into two categories, namely, static and dynamic DTs. The former type of digitized models neither change in shape nor affected by data streams. However, the latter type of DTs keeps changing based on the updated streams of data. They capture live performances of the real objects. Thus, they always stay updated and can be altered instantaneously.

Since multidisciplinary teams are involved in creating DTs, the interaction between the teams, workflows, and progresses need to be monitored in a trustworthy manner. Each collaboration activity that occurs between different phase providers, engineers, and managers must be documented in such a way that it ensures transparent history monitoring, traceability, privacy, trust, and security [8]. This can be achieved using blockhain that is based on decentralized and distributed ledger [9], [10]. Blockchain ensures provenance data tracking and tracing on-chain [11]. It also provides other features that include trust, accountability, data integrity, and immutability which make it an ideal solution to monitor the creation process of DTs. Blockchain allows the exchange of timestamped events and notifications which are permanently stored in a secure and tamper-proof ledger [12]. In sum, the pairing of DTs with blockchain ensures secure, efficient, decentralized, and trusted creation of virtual models. The creation process usually involves four phases, e.g., design phase, building phase, testing phase, and delivery phase [8]. The beginning of each phase depends on the completion of the previous phase. These phases can further be divided based on hardware design, quality testing, dimensional analysis, and others [6]. Current creation processes of DTs are mostly based on simple tools, thereby ended up using centralized solutions that are vulnerable to single point of failures. However, incorporation of blockchain ensures that DTs creation processes are managed in a secured and trusted manner.

A. RELATED WORKS AND CONTRIBUTIONS

DTs are still in their infancy and as such very limited literature is available on the subject. Herein, we discuss the existing solutions concerning to creation process of DTs and their convergence with blockchain.

In most cases, multidisciplinary teams having extensive experience are involved to create the flexible, scalable, resilient to operational changes, and expandable DTs [3]. However, the current creation of DTs is based on traditional methods that have centralized authorities to manage the truth and facts [3], which do not ensure trustworthiness. Authors of [13] have discussed the challenges that hinder the successful and secure creation of DTs. First, the interoperability between the different collaborators and associates is considered as an indispensable issue. Second, data storage and analysis are other aspects that need to be considered. Third, ensuring real-time communication transparency between the contributors is a vital feature that must be incorporated in the creation process of DTs [13].

The study conducted in [14] presents important characteristics (e.g., data provenance tracking, traceability, transparency, and tamper-proof logs) of blockchain that make it best fit for supply chain industries. These characteristics also make blockchain a highly secure distributed ledger that is trusted throughout the end-to-end creation process of DTs. Blockchain acts as a unity layer where it brings together all the transactions and interactions of the stakeholders into transparency and accountability [8]. Authors of [8] have introduced the concept of employing DTs for additive manufacturing in the aircraft industry. The authors proposed a theoretical solution that highlights the importance of blockchain to secure the data associated with the aircraft industry. They used visual studio tool to write and test their Javascript code [8].

Motivated by the need of a secure, trusted, resilient and reliable method to track and trace the different phases involved in the creation of DTs, this study aims to design and implement a decentralized blockchain-based solution. Unlike the existing works, the key contributions of our paper can be summarized as follows:

- We propose a blockchain-based creation process of DTs that guarantees secure and trusted traceability, accessibility, and immutability of transactions, logs, and data provenance.
- We introduce smart contracts to govern and track transactions initiated by participants involved in the creation of DTs.
- We integrate our blockchain-based system with the InterPlanetary File System (IPFS) to store and share information of DTs.

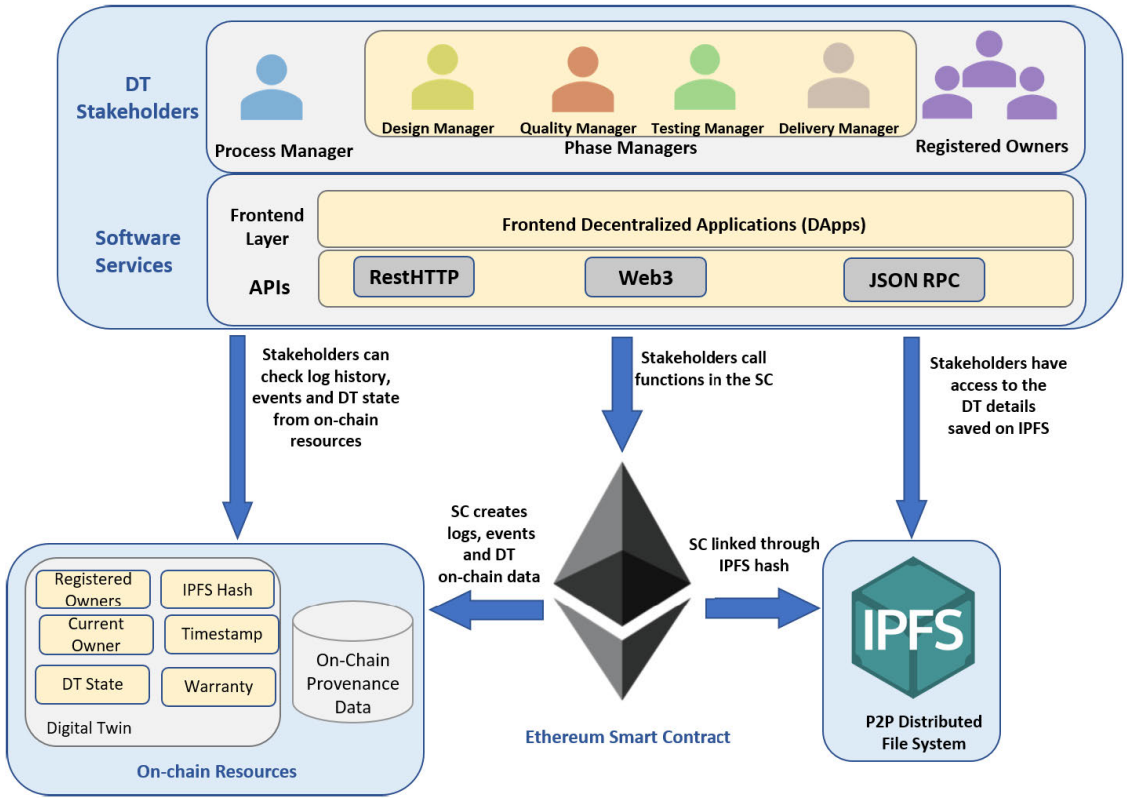


FIGURE 2. A system diagram showing the main components interacting with the smart contract and the resulting on-chain resources.

- We provide security and cost analysis, and show how our approach fulfills the requirements of DTs process creation.
- We propose a generic solution that can be customized to fulfill the needs of any industry.
- We present the full implementation details, smart contract code,[1] and testing details.

The remainder of this paper is organized as follows. Section II presents the proposed blockchain-based approach. Section III describes the implementation details. Section IV provides testing and validation details. Section V evaluates the implemented solution, and section VI concludes the paper.

II. PROPOSED BLOCKCHAIN-BASED SOLUTION

This section describes the proposed blockchain-based approach for the creation process of DTs. This is a generic approach that can be customized to fulfill the requirements and functionalities of any company that needs a digital model to meet certain objectives.

To create a DT, which is a digitized virtual model of a real physical object, multidisciplinary teams gather up and work together. The DT creation process requires trusted management and tracking to assure that entire history information is kept in a tamper-proof manner. Thus, blockchain is employed to meet strict security requirements in a decentralized way. The components of the proposed solution are presented in figure 2. In the figure, the DT stakeholders, i.e., process managers, phase managers, and owners interact with the smart contract through a front-end layer using Application Program Interfaces (APIs). The front-end decentralized application may use any of the interfaces, such as RestHTTP, Web3 or JSON RPC to connect the stakeholders to the smart contract or IPFS servers as shown in 2. The interactions with the smart contract ensure secure on-chain resources that are traceable and tamper-proof. Further details of different system components are provided below.

- **DT Stakeholders:** The stakeholders include process manager, design manager, quality manager, testing manager, delivery manager, and registered DT owners. These stakeholders act as participating entities in the smart contract. They are authorized to execute certain function calls depending on the state of the DT. Moreover, they can access the on-chain resources, such as the provenance data and DT details to keep track of the DT state, history and logs information. Furthermore, they have authorized access to the full information of the DT stored on the IPFS servers [15].
- **IPFS Storage:** We employ IPFS to store the required details of DTs or agreement forms in a decentralized way. The IPFS-based storage ensures reliability, accessibility, and integrity of the stored data. Moreover,

[1] https://github.com/smartcontract694/DT/blob/master/code

the costs of storing on IPFS is very minor compared to storing on-chain. Additionally, the cost of storing on IPFS would not cost more than conventional storage spaces and it is a fixed monthly amount that depends on the amount of storage space required.
IPFS ensures data integrity through the IPFS hash which is uniquely generated for every file uploaded on the IPFS servers. All the information and data of the DT creation process are stored on IPFS and their unique hash is stored in the smart contract. If anyone tries to modify the information on the IPFS servers, the newly generated hash of the file would not match the hash stored in the smart contract. Hence, storing the data on IPFS ensures that the data is securely stored with high integrity [16].

- **Ethereum Smart Contract:** The concept of Ethereum smart contract is introduced to manage the creation process of DTs. The contract facilitates in terms of logistics tracking and manages all the history of transactions. The contract also deals with IPFS hash that leads to accessing the DT information from IPFS servers.
- **On-chain Resources:** Since all the process creation phases are monitored through the smart contract, thereby creating a large number of transaction logs. Storing such information on-chain creates an important resource for tracing and tracking, and making all stakeholders accountable for their actions. On-chain resources also handle certain types of DT information, such as the timestamp, owner, registered owners, state, IPFS hash, and warranty.

A. PERMISSIONED AND PERMISSIONLESS BLOCKCHAIN

The proposed solution is based on the Ethereum blockchain platform. By design, Ethereum is a permissionless public blockchain. However, a private or permissioned blockchain can be used if data privacy is a key issue, to restrict access to the blockchain. The presented architecture, techniques, algorithms, and code are generic and could be used on both permissioned and permissionless blockchain platforms.

A permissioned blockchain can be built with Ethereum by setting up a private instance of the Ethereum blockchain (nodes, wallets, and funds in private control). However, Ethereum blockchain does not support complex privacy and permissioning features by design. Thus, a more suitable approach would be to use a blockchain with such features already built-in, such as Hyperledger Fabric (HF), which is designed as a permissioned private blockchain, where only specific entities/users are authorized to use, validate, and access the blockchain [17], [18]. HF relies on a crash-fault tolerant, decentralized and deterministic consensus algorithm, guaranteeing finality and correctness of blocks. As a result, ledgers cannot fork, performance and scalability are increased, and transaction throughput is significantly superior compared to proof-of-Work (PoW) consensus used in Ethereum [19]. Most important components of a HF blockchain are the organizations, peers, ordering service, membership service providers (MSPs), channels and chaincodes. Organizations refer to business entities participating in the network: they can have access to multiple channels and can issues identities to network participants through their associated MSP. MSPs offer an abstraction layer for a membership orchestration architecture. HF's certificate authority component is the default implementation of MSP interface that includes identity management operations such as certificate issuance, renewal and revocation. Channels are components that enable transaction confidentiality in HF. They represent physically separate, independent and private instances of blockchain ledger visible only to organizations that are members of the channel. All data associated with private channels remain inaccessible and invisible to any unauthorized network organizations. Peers are components that communicate with clients on one side and are responsible for committing blocks to the world state on the other. Each peer holds its copy of ledgers, and an organization can have multiple peers (i.e. for redundancy). Consensus in networks is achieved through the Ordering Service. It is a decentralized service offering fast ordering of transactions into blocks. The ordering service also enforces basic access control for channels, restricting who can read and write data to them, and who can configure them. Finally, chaincodes provide the means to implement business functionalities in form of smart contracts (in Golang, Java, or NodeJS).

If Ethereum-like blockchains are favored, Hyperledger Besu is an option to create permissioned private blockchain networks. Besu is an open-source Ethereum client that can be used to build a permissioned private Ethereum blockchain. Besu nodes can connect to Ethereum public (MainNet), or test networks (Ropsten, RinkeBy, Kovan, etc.). Alternatively, one can create a private Besu network with private nodes, wallets, smart contracts, and funds. Being well ahead of Ethereum when it comes to privacy and permissioning features offering, Hyperledger Besu provides smart contract-based permissioning, account and node-level whitelisting, high transaction throughput consensus mechanisms (IBFT, Clique PoA) and private transactions and privacy groups [20]. Moreover, Hyperledger Besu can use a separate component for managing identities and private keys, such as 'EthSigner' [21]. Access control and management to support private transacting are provided by 'Orion' which is also Besu's private transactions manager [21], [22].

B. PHASES INVOLVED IN THE DT CREATION PROCESS

Figure 3 shows the main phases involved in the creation process of DTs. There are mainly four phases:

- **Design:** Using Computer-Aided Design (CAD) Tools, the design engineers apply modeling techniques and analyze the digital data. The purpose is to directly capture objectives of the abstract model and convert it into a virtual replica of the real figure.
- **Build:** The virtual design is transformed into a real model using sensory data. This is a critical phase where the model is updated continuously from captured data

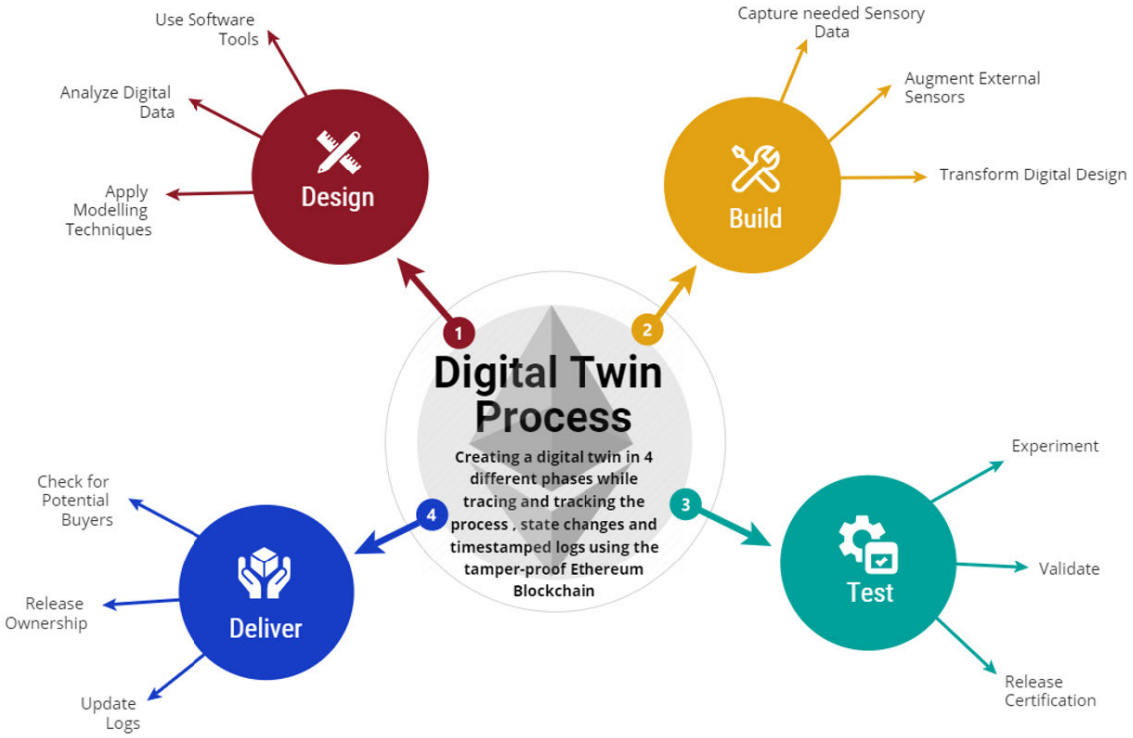


FIGURE 3. The four main phases involved in the DT creation process using blockchain as the managing entity.

and sends feedback of analytical information. It is essential to ensure that the build model works well in the real environment whether it is in a supply chain industry or any other heavily hectic data-dependent circumstances.

- **Test:** When the model is built successfully, it is tested using a test bench to exclude any logical or possible design defects in the DT model. Validation and compliance with the certification standards are obligatory steps to meet the quality assurance and control standards. Certain thresholds of acceptance must be met to provide confidence that certain requirements are fulfilled.
- **Deliver:** After successful testing and validation, the DT model is ready for deployment. Potential owners can register and the ownership rights are released to them by the current DT owner. In this proposed design, all the owners are available on the blockchain.

III. IMPLEMENTATION DETAILS

We used Solidity language to write the smart contract which was compiled and tested using the Remix IDE [23]. Solidity is the most popular language used for writing Ethereum smart contracts. A debugger is embedded within Remix to check for any warnings and fix possible errors. This section briefly explains what functions are used in the code and how they are implemented. Also, the code is publicly made available on GitHub.

The process manager is responsible for initiating the creation process of DTs. Hence, the process manager is the smart contract owner. The process starts after an event is triggered that notifies all the other managers (participating entities). This event triggers a flow of function calls that starts with the design manager and ends with the delivery manager. All these calls are logged and all the transactions can be traced back when needed. Each function call produces an event and each phase has a beginning and an end.

Modifiers are used to restrict access on who can execute a function call. This is done using the Ethereum addresses of the participating entities. Therefore, only authorized Ethereum addresses can make a function call. If for any reason, an unauthorized entity tries to execute a function, the contract state is reverted to its original state. Hence, the DT state is not amended or affected in anyway. This is captured in the algorithms using the statement 'Revert Contract State', which happens if any of the function restrictions and requirements are not met.

Each phase has two main functions. One is responsible to commence a new phase and the other is to approve its results. The DT can be modified in a phase between those two function calls as much as needed. The changes in between the two function calls are not captured on-chain. However, if a certain use case requires capturing changes during a phase, a new function can be added. The current implementation is a generic one that can be modified based on the requirements and use case. The state of the DT keeps changing on-chain as the phases proceed and as the functions get called. Once all required steps of a phase are completed, the phase manager executes the last function call which would change the DT state and end the current phase. The states and roles are used

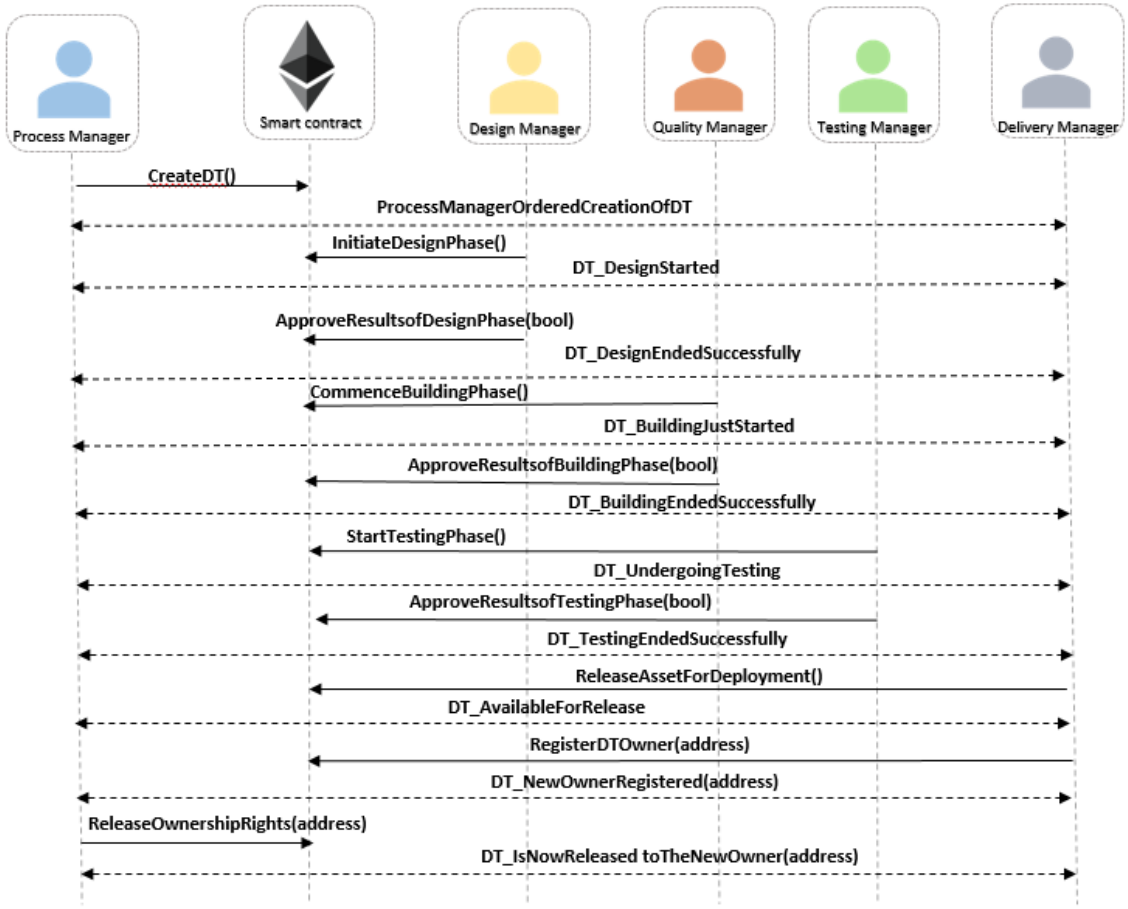


FIGURE 4. Sequence diagram showing all the interactions between the participating entities of the smart contract.

to ensure the correct flow of function calls which determines the process sequence.

Several owners can own the DT. When the DT is ready for deployment, the delivery manager releases the asset for deployment and a new owner gets registered by the delivery manager. Then the process manager who currently owns the DT releases the ownership rights to the new registered owner. Later on, the registered owner can announce that the DT is available for release to another new owner. All the Ethereum addresses of the DT owners can be traced and tracked using the events available in the logs. However, the detailed information of the DT model and data is stored in a decentralized way using the IPFS servers [15] because storage on the blockchain is highly expensive. Furthermore, all the participating entities agree to fulfill their roles based on the company's terms, conditions, and regulations. This agreement form is also saved on the IPFS server.

Figure 4 illustrates the interactions between the participating entities from the start to the end of the creation process. Those are the main interactions along with their subsequent events. However, other events are also triggered in the case of disapproval of the results by any of the managers. Each phase can be further divided as required by the DT model. For instance, if additional managers or collaborators are needed on-chain the same existing criteria can be followed. Any task can be handled by a certain additional entity using two functions. Moreover, we have shown how a superior manager needs to approve the tasks of the team by showing the relationship between the engineer and the line manager. The same could be applied to customize any phase depending on the use case. Further details of each function call are provided in their algorithms that can be found in the following subsections.

1) INITIATING THE PROCESS OF CREATING THE DT

Algorithm 1 describes how the process of creating the DT is started. The process manager starts the process which triggers an event that leads to a change in the DT state to 'Ordered to be Created'. This allows the first phase of the process to take place. If anyone else tries to initiate the process other than the process manager, the contract shows an error and the state of the contract is reverted to its original state.

2) STARTING THE DESIGN PHASE

Algorithm 2 describes the design phase initiated by the authorized design manager by changing the state of the DT to 'In

Algorithm 1 Creating the Digital Twin

```
Input : caller, DTstate, Process Manager
caller is the Ethereum Address of the function caller.
if caller == ProcessManager then
    if DTstate == NotCreated then
        DTstate = OrderedToBeCreated.
        Create a notification about the order by the
        Process Manager to initiate the creation process.
    end
    else
        Revert contract state and show an error.
    end
end
else
    Revert contract state and show an error.
end
```

Algorithm 2 Initiating Design Phase

```
Input : caller, DTstate, Design Manager
caller is the Ethereum Address of the function caller.
if caller == DesignManager then
    if DTstate == OrderedToBeCreated then
        DTstate = InDesignPhase.
        Create a notification about the start of the design
        phase.
    end
    else
        Revert contract state and show an error.
    end
end
else
    Revert contract state and show an error.
end
```

Design Phase'. This creates a notification and sends it to all the participating entities to let them know that the design phase has started.

3) APPROVING THE RESULTS OF THE DESIGN PHASE

Once the design phase is completed, the design manager checks the details and signs with approval, so the next phase can be started. This approval of results is carried out using a boolean as explained in algorithm 3. This changes the state of the DT to 'Done with Design Phase' and triggers an event to announce it. If the design manager disagrees with the results for any reason or faced issue, the state of the DT is not updated.

4) COMMENCING THE BUILDING PHASE

Algorithm 4 describes how the building phase starts with the permission of the quality manager. This triggers a notification that announces to the listeners to change the status of DT to 'In Building Phase'. This action of starting the new building phase can only occur if the results of the design

Algorithm 3 Approving Results of the Design Phase

```
Input : caller, DTstate, Design Manager, result
caller is the Ethereum Address of the function caller.
if caller == DesignManager then
    if DTstate == InDesignPhase then
        if result == True then
            DTstate = DoneWithDesignPhase.
            Create a notification about the end of a
            successful design phase.
        end
        else
            Create a notification stating that the next
            phase cannot be started.
        end
    end
    else
        Revert contract state and show an error.
    end
end
else
    Revert contract state and show an error.
end
```

Algorithm 4 Commencing Building Phase

```
Input : caller, DTstate, Quality Manager
caller is the Ethereum Address of the function caller.
if caller == QualityManager then
    if DTstate == DoneWithDesignPhase then
        DTstate = InBuildingPhase.
        Create a notification about the start of the
        building phase.
    end
    else
        Revert contract state and show an error.
    end
end
else
    Revert contract state and show an error.
end
```

phase have been successfully approved by the design manager and the previous state of the DT is 'Done with Design Phase'.

5) APPROVING RESULTS OF THE BUILDING PHASE

It is important for the quality manager to carefully check the results of the building phase before starting the testing phase. This is ensured using algorithm 5, where only the quality manager is allowed to update the DT state to 'Done with Building Phase'. This in return triggers an event that yields in all the listeners knowing that the building phase is completed. If the produced results were not as per expectations of the quality manager, then the signature of the manager can be 'False' to indicate that the DT state cannot be updated that

Algorithm 5 Approving Results of the Building Phase

```
Input : caller, DTstate, Quality Manager, result
caller is the Ethereum Address of the function caller.
if caller == QualityManager then
    if DTstate == InBuildingPhase then
        if result == True then
            DTstate = DoneWithBuildingPhase.
            Create a notification about the end of a
            successful building phase.
        end
        else
            Create a notification stating that the next
            phase cannot be started.
        end
    end
    else
        Revert contract state and show an error.
    end
end
else
    Revert contract state and show an error.
end
```

Algorithm 6 Starting Testing Phase

```
Input : caller, DTstate, Testing Manager
caller is the Ethereum Address of the function caller.
if caller == TestingManager then
    if DTstate == DoneWithBuildingPhase then
        DTstate = InTestingPhase.
        Create a notification about the start of the testing
        phase.
    end
    else
        Revert contract state and show an error.
    end
end
else
    Revert contract state and show an error.
end
```

requires to move on the next phase. Subsequently, the issue can be resolved off the chain, and then the design manager can sign for approval.

6) STARTING THE TESTING PHASE

Once the quality manager approves the results of the design phase, the testing manager initiates the testing phase. This changes the DT state to 'In Testing Phase' as shown in algorithm 6. This creates an event to inform all listeners about the beginning of the testing phase. If any other Ethereum address tries to start the testing phase, the contract state will revert to its previous state by showing an error.

Algorithm 7 Approving Results of the Testing Phase

```
Input : caller, DTstate, Testing Manager, result
caller is the Ethereum Address of the function caller.
if caller == TestingManager then
    if DTstate == InTestingPhase then
        if result == True then
            DTstate = DoneWithTestingPhase.
            Create a notification about the end of a
            successful testing phase.
        end
        else
            Create a notification stating that the next
            phase cannot be started.
        end
    end
    else
        Revert contract state and show an error.
    end
end
else
    Revert contract state and show an error.
end
```

7) APPROVING RESULTS OF THE TESTING PHASE

When the testing phase is completed, the testing manager checks the results and confirms by signing so that the next phase can be started. This procedure is performed using algorithm 7 where the testing manager uses a Boolean to approve continuing of the next phase. The state of the DT gets updated accordingly.

8) RELEASING THE DT FOR DEPLOYMENT

Once the testing phase is completed, the DT is ready to be deployed. Therefore, the delivery manager creates an event after changing its state to "Available for Release". This procedure is shown in algorithm 8 that explains how it can be executed by the delivery manager when testing of the DT is completed. When an owner decides that the DT can be released to another prospective new owner, this algorithm can also be executed by the current owner of the DT.

9) REGISTERING THE DT OWNERS

Once the DT is available for release to its new owner, the delivery manager registers the new owner by announcing the Ethereum address of the new owner to all the participating entities using an event in the logs file. This is an important step to keep tracking of all registered owners of the DT model. Algorithm 9 shows how the state gets updated to 'Reserved for New Owner' when the new owner's Ethereum address is successfully announced.

10) RELEASING OWNERSHIP RIGHTS

The final step is releasing the ownership rights which are explained briefly in algorithm 10. Releasing the ownership rights can only be performed by the DT owner. When the

Algorithm 8 Releasing Asset for Deployment

Input : caller, DTstate, Delivery Manager, DTcurrentOwner

caller is the Ethereum Address of the function caller.
if *(caller == DeliveryManager ∧ state == DoneWithTestingPhase) ∨ (caller == DTcurrentOwner ∧ state == ReleasedToNewOwner)* **then**
    DTstate = AvailableForRelease.
    Create a notification about the availability of the DT for release by the caller.
end
else
    Revert contract state and show an error.
end

Algorithm 9 Registering DT Owner

Input : caller, DTstate, Delivery Manager, newOwner, DTcurrentOwner

caller is the Ethereum Address of the function caller.
newOwner holds the Ethereum Address of the DT's new owner.
if *caller == DeliveryManager* **then**
    if *DTstate == AvailableForRelease ∧ newOwner ≠ DTcurrentOwner* **then**
        DTstate = ReservedForNewOwner.
        Create a notification about the newly registered owner of the DT using address *newOwner*.
    end
    else
        Revert contract state and show an error.
    end
end
else
    Revert contract state and show an error.
end

first time this algorithm is executed, then the process manager is the only authoritative entity that can release the ownership rights to the first official owner. As can be seen in this algorithm, we have avoided using lists or arrays to store the registered owners on the chain as it is expensive. Instead, we are depending on a blockchain that uses only one attribute to hold the Ethereum address of the DT owner. All owners' history can be traced back using the emitted events which hold the Ethereum address of every owner.

Our implementation code can be customized to any application that requires a DT. It is not specific to a certain industry or manufacturing model. Moreover, the phases can include more specific specialties based on the desired requirements. Table 1 describes the functions used in the code along with their descriptions.

Algorithm 10 Releasing Ownership Rights

Input : caller, DTstate, DTcurrentOwner, newOwner

caller is the Ethereum Address of the function caller.
DTcurrentOwner is the Ethereuem Address of the DT's owner.
newOwner is the Ethereum Address of the new DT owner.
if *caller == DTcurrentOwner* **then**
    if *DTstate == ReservedForNewOwner* **then**
        DTcurrentOwner = newOwner
        DTstate = ReleasedToNewOwner.
        Create a notification about the DT being owned now by its new owner.
    end
    else
        Revert contract state and show an error.
    end
end
else
    Revert contract state and show an error.
end

FIGURE 5. Logs showing a successful request to create the DT by the process manager.

IV. TESTING AND VALIDATION

Herein, we discuss testing and validation of the smart contract. The functions of the smart contract and their logical flows were tested using Remix IDE. There are five participants interacting with the smart contract. Each of the participating entities has an Ethereum address. For example, the process manager has "0xCA35b7d915458EF540aDe6068dFe2F44E8fa733c", the design manager has "0 × 14723A09ACff6D2A60DcdF7aA4AFf308FDDC160C", the quality manager has "0 × 4B0897b0513fdC7C541B6d9D7E929C4e5364D2dB", the testing manager has "0×583031D1113aD414F02576BD6afaBfb302140225", and the delivery manager has"0xdD870fA1b7C4700F2BD7f44238821C26f7392148". Each of the functions in the smart contract is associated with a DT state. Moreover, only specific roles are allowed to execute each function. If either of these criteria is not met, the contract state reverts back to its original state. Each algorithm is tested and the results are shown in a figure associated with each test

TABLE 1. Functions along with their Descriptions.

Function	Description
constructor()	Initializes the DT parameters, state, IPFS Hash, and managers' and owner's Ethereum addresses.
CreateDT()	Updates the DT state by the process manager to 'OrderedToBeCreated'.
InitiateDesignPhase()	Updates the DT state to 'InDesignPhase' by the design manager.
ApproveResultsofDesignPhase(bool)	Based on the boolean parameter the DT state is either updated to 'DoneWithDesignPhase' by the design manager or remains unchanged.
CommenceBuildingPhase()	Updates the DT state to 'InBuildPhase' by the quality manager.
ApproveResultsofBuildingPhase(bool)	Based on the boolean parameter the DT state is either updated to 'DoneWithBuildingPhase' by the quality manager or remains unchanged.
StartTestingPhase()	Updates the DT state to 'InTestingPhase' by the testing manager.
ApproveResultsofTestingPhase(bool)	Based on the boolean parameter the DT state is either updated to 'DoneWithTestingPhase' by the testing manager or remains unchanged.
ReleaseAssetForDeployment()	The delivery manager or the current owner can decide to update the DT state to 'AvailableForRelease'.
RegisterDTOWner(address)	This function takes the address of the new DT owner and announces it to the participating entities. It also updates the DT state to 'ReservedForNewOwner'.
ReleaseOwnershipRights(address)	Here the state is updated to 'ReleasedToNewOwner' after the function changes the Ethereum address of the current owner to the address of the new Owner. This function is accessible to current DT owner only.

case. The figures are used to show the hash, events triggered, and the function executed.

A. INITIATING THE DT CREATION PROCESS

The process manager is authorized to start the DT creation process using the function named *CreateDT*(). Figure 5 shows the logs which are produced after the DT has successfully been updated and an event is triggered to let other participants to take actions accordingly.

B. COMMENCING AND APPROVING THE DESIGN PHASE

The design manager commences the design phase and updates the DT state successfully using the function named *InitiateDesignPhase*(). The design manager also approves

FIGURE 6. Logs presenting a successful beginning and end of the design phase by the design manager.

FIGURE 7. Logs showing successful starting and ending of the build phase by the quality manager.

the results of the design process using the function named *ApproveResultsofDesignPhase*, as shown in figure 6. The figure shows how a successful event is triggered based on the 'True' boolean result passed by the design manager. Both functions show errors if executed by any unauthorized Ethereum Address.

C. STARTING AND ENDING THE BUILD PHASE

The quality manager starts the build phase by calling a function named *CommenceBuildingPhase*. The function changes the DT state and notifies the listeners about the beginning of the new phase, as shown in figure 7. The quality engineer also ends the phase by approving and signing on the results using the function named *ApproveResultsofBuilding Phase*.

D. BEGINNING AND FINISHING THE TESTING PHASE

The last step prior to the deployment phase is final testing and validation. This is done by the authorized testing engineer by executing the function named *StartTestingPhase*.

FIGURE 8. Logs showing a successful beginning and end to the testing phase as approved by the testing manager.

FIGURE 9. The DT was successfully released for deployment for the first time by the delivery manager after a successful testing phase.

Subsequently, a notification is triggered to inform everyone that the DT has been successfully tested, as shown in figure 8. This is done by executing the function named *ApproveResultsofTestingPhase*.

E. RELEASING ASSET FOR DEPLOYMENT

The successful testing of the DT makes it initially ready for the deployment. The delivery manager makes a change in the DT status by executing the function named *ReleaseAsseForDeployment* which creates a notification, as shown in figure 9.

This function can also be executed by the owner of the DT if the state of the DT was ''ReleasedToNewOWner''. The state ''ReleasedToNewOWner'' indicates that the DT is now owned by an owner that can release it for deployment. This can be seen in figure 10, where the owner with Ethereum address ''0xba1d1ffc6188e365c70592083a5213d480db7451'' releases the asset for deployment.

F. REGISTERING A NEW DT OWNER

When a new owner for the DT is available, the delivery manager uses the function named *RegisterNewOwner* to register the Ethereum address of the new owner in the logs. In this example, the new owner's Ethereum address is

FIGURE 10. The DT owner successfully released it for deployment.

FIGURE 11. The new DT owner has successfully been announced in an event in the logs.

''0xba1d1ffc6188e365c70592083a5213d480db7451''. This function is executed successfully, as can be seen in figure 11.

G. RELEASING OWNERSHIP RIGHTS

After a successful registration of the new DT owner, the current owner which is in our case a process engineer can only release ownership rights. This is the final step that results in changing the current ownership parameter in the DT structure defined on the contract. Hence, the new owner thereafter can release the asset for deployment. In figure 12, the owner which in this case was the process manager, releases the ownership rights to the registered DT owner with Ethereum address ''0xba1d1ffc6188e365c70592083a5213d480db7451''.

V. EVALUATION

The proposed blockchain-based approach is evaluated to ensure that it meets the DTs creation requirements in terms of trust and security.

A. SECURITY ANALYSIS

The implementation code has successfully been tested and cross-checked using a security tool named 'SmartCheck' [24], [25]. The tool evaluates the solidity code against any vulnerabilities or even bad practices, such as reentrancy, timestamp dependence, denial of service (DoS), locked money or costly loops, and others. The security analysis results reveal that no known vulnerabilities were found. Only specific errors were categorized in the tool as 'bad practices' and they have been used in our code for testing purposes. For example, the hardcoded addresses of the stakeholders will

FIGURE 12. The new DT owner has successfully been updated as the ownership rights were released by the former owner.

not be needed in the real solution. Moreover, the compiler version can be fixed as per the application needs. In a nutshell, the code is free from any recognizable bugs, and is reliable and maintainable.

B. COST ANALYSIS

Cost analysis of each on-chain transaction is important as it affects the reliability and feasibility of the solution. The cost of the functions in the contract is almost negligible as the functions mainly change the DT state and create a notification about the current phase state. Also, in our implementation, we relied on the logs to save the owner information of the DT. Thus, we avoided using arrays or mappings as their costs are much higher compared to storing only the current owner's Ethereum address. Consequently, a smart contract is responsible for the creation and tracking of one DT. Therefore, multiple smart contracts are needed for multiple DTs. This is more convenient, unless the owners prefer otherwise for logistics reasons then mappings can be used in the code to map different DT structures to managerial structures. In our implementation, we relied on a single DT to better present the idea as well as to avoid unnecessary costs of using structures and mappings which could only be needed for certain justifiable use cases. Also, in our implementation of only one DT per smart contract, we avoided additional costs to ensure that our solution is feasible.

Table 2 shows the storage cost of the DT owners in an array as the number of registered owners increases. The ETH Gas Station was used to find the cost in dollars of each transaction [26]. The average gas price which is three Gwei was used at the time of finding the cost in USD. The fastest, fast, average and cheap prices are 11, 10, 3 and 1 Gwei, respectively. We have used the average price in computing the values and presented the cost in US dollars as can be found in the tables 2-3 which show that the cost of storing in an array is much higher. This is because storing in an array also involves storing the length of the elements. Moreover, both tables 2-3 indicate that first time storage costs are the highest. This factor indicates that changing the values of a storage space from zero (non-initialized) to non-zero costs higher regardless of the type of storage. Subsequent additions to the array all cost the same which is less than the first time. In addition, storing in an array the first time involves storing

TABLE 2. Storage costs of registered owners in an array.

Number of Owners	Transaction Gas	Execution Gas	Total Gas	Cost ($)
1	70577	47961	118538	0.062
2	55577	32961	88538	0.046
3	55577	32961	88538	0.046

TABLE 3. Storage costs of current owner attribute.

Number of Executions	Transaction Gas	Execution Gas	Total Gas	Cost ($)
1	35341	12661	48002	0.0252
2	35277	12661	47888	0.02515

the element as well as the length. Hence, a store operation which costs 20K is actually doubled for an array when the storage value is set to non-zero from zero [27]. Consequently, this storage operation for an array would cost at least 40K. This is double the cost needed for single storage spaces where it costs only 20K the first time.

It can also be seen from the tables that in the case of array the storage cost of the first time is 1.6% higher than the next subsequent times. However, the difference between the cost of storing in a single storage space by first time and the subsequent times is almost negligible, only 0.00005$. Moreover, the cost of storing in an array by the first time is 3.68% higher and for the subsequent times it is 2.09% compared to a single storage space.

C. SATISFYING DT REQUIREMENTS

To secure the creating process of DTs, our solution uses blockchain to ensure history tracking and traceability. These features are ensured using tamper-proof logs. These logs help to ease and secure the look-up process and expedite problem-solving, process management, and traceability during the creation process of DTs. Data integrity is one of the important features of blockchain. All transactions have a hash and they are timestamped. Therefore, on-chain transactions are safe from replay and Man in the Middle (MITM) attacks, thereby making them immutable.

Furthermore, the framework's security relies on the intrinsic features of blockchain, which also include non-repudiation. The Ethereum addresses of the participating entities are used to sign each transaction as they possess unique asymmetric keys. Therefore, ensuring non-repudiation and accountability. The proposed solution employs IPFS to store and share the DTs data. The proposed solution ensures non-repudiation and accountability of each DT creation phase by using restricted function calls in the Ethereum smart contract. Therefore, each phase manager and DT owner are accountable for their actions. Moreover, each transaction is signed by its initiator, which is later on saved as part of the reliable logs. Additionally, the creation process of DTs using blockchain is reliable and resilient to security vulnerabilities. The on-chain provenance data is always accessible. In addi-

tion, the functions of the smart contract are flexible and can be amended to meet the specific needs of any industry.

VI. CONCLUSION

In this paper, we have designed and implemented a blockchain-based creation process for DTs in a manner that is decentralized, tamper-proof, immutable, and secure. Our design approach eases the management of the process from the design phase all the way to the delivery and release of ownership rights. It also keeps track of all the registered owners and provides a history of the process on-chain. In our approach, we made use of the decentralized IPFS storage servers to store the details of the DTs. We also provided a solution framework, detailed implementation and testing results. In addition, the smart contract code is publicly made available on GitHub and is generic enough to be customized for the use case of each enterprise as it requires. The provided code meets the security requirements and has been analyzed successfully using the SmartCheck tool. Moreover, it is secure against the commonly known security vulnerabilities and attacks. Cost analysis was also discussed as part of the evaluation. As a future work, we plan to implement a complete solution composed of private blockchain nodes in addition to developing frontend decentralized apps (DApps) to be used by different participants. For our private blockchain platform, we will consider the use of HF and Hyperledger Besu.

REFERENCES

[1] N. Mohammadi and J. E. Taylor, "Smart city digital twins," in *Proc. IEEE Symp. Ser. Comput. Intell. (SSCI)*, Honolulu, HI, USA, Nov. 2017, pp. 1–5.

[2] F. Tao, J. Cheng, Q. Qi, M. Zhang, H. Zhang, and F. Sui, "Digital twin-driven product design, manufacturing and service with big data," *Int. J. Adv. Manuf. Technol.*, vol. 94, nos. 9–12, pp. 3563–3576, Mar. 2017.

[3] B. Schleich, N. Anwer, L. Mathieu, and S. Wartzack, "Shaping the digital twin for design and production engineering," *CIRP Ann.*, vol. 66, no. 1, pp. 141–144, 2017.

[4] A. El Saddik, "Digital twins: The convergence of multimedia technologies," *IEEE MultimediaMag.*, vol. 25, no. 2, pp. 87–92, Apr. 2018.

[5] M. Grieves and J. Vickers, "Digital twin: Mitigating unpredictable, undesirable emergent behavior in complex systems," in *Transdisciplinary Perspectives on Complex Systems: New Findings and Approaches*. Berlin, Germany: Springer, 2017, pp. 85–113.

[6] R. Stark, C. Fresemann, and K. Lindow, "Development and operation of digital twins for technical systems and services," *CIRP Ann.*, vol. 68, no. 1, pp. 129–132, 2019.

[7] *Digital Twin Technology Benefits and Challenges*. Accessed: Dec. 12, 2019. [Online]. Available: https://www.identitymanagement-institute.org/digital-twin-technology-benefits-and-challenges/

[8] C. Mandolla, A. M. Petruzzelli, G. Percoco, and A. Urbinati, "Building a digital twin for additive manufacturing through the exploitation of blockchain: A case analysis of the aircraft industry," *Comput. Ind.*, vol. 109, pp. 134–152, Aug. 2019.

[9] A. Dorri, S. S. Kanhere, R. Jurdak, and P. Gauravaram, "Blockchain for IoT security and privacy: The case study of a smart home," in *Proc. IEEE Int. Conf. Pervas. Comput. Commun. Workshops (PerCom Workshops)*, Kona, HI, USA, Mar. 2017, pp. 618–623.

[10] H. R. Hasan and K. Salah, "Combating deepfake videos using blockchain and smart contracts," *IEEE Access*, vol. 7, pp. 41596–41606, 2019.

[11] P. Ruan, G. Chen, T. T. A. Dinh, Q. Lin, B. C. Ooi, and M. Zhang, "Fine-grained, secure and efficient data provenance on blockchain systems," *Proc. VLDB Endowment*, vol. 12, no. 9, pp. 975–988, May 2019.

[12] K. Christidis and M. Devetsikiotis, "Blockchains and smart contracts for the Internet of Things," *IEEE Access*, vol. 4, pp. 2292–2303, 2016.

[13] S. P. A. Datta, "Emergence of digital twins," 2016, *arXiv:1610.06467*. [Online]. Available: https://arxiv.org/abs/1610.06467

[14] H. M. Kim and M. Laskowski, "Toward an ontology-driven blockchain design for supply-chain provenance," *Intell. Syst. Accounting, Finance Manage.*, vol. 25, no. 1, pp. 18–27, Mar. 2018.

[15] *IPFS is the Distributed Web*. Accessed: Jun. 13, 2018. [Online]. Available: https://ipfs.io/

[16] N. Nizamuddin, H. R. Hasan, and K. Salah, "IPFS-blockchain-based authenticity of online publications," in *Proc. Int. Conf. Blockchain*. Cham, Switzerland: Springer, 2018, pp. 199–212.

[17] L. Severeijns. (2017). *What is Blockchain? How is it Going to Affect Business?* [Online]. Available: https://www.math.vu.nl/ sbhulai/papers/paper-severeijns.pdf

[18] E. Androulaki, A. Barger, V. Bortnikov, C. Cachin, K. Christidis, A. De Caro, D. Enyeart, C. Ferris, G. Laventman, and Y. Manevich, "Hyperledger fabric: A distributed operating system for permissioned blockchains," in *Proc. 13th EuroSys Conf.*, Porto, Portugal, 2018, pp. 1–15.

[19] C. Harris, "Improving telecom industry processes using ordered transactions in hyperledger fabric," in *Proc. Global Commun. Conf.*, Waikoloa, HI, USA, 2019, pp. 1–6.

[20] *Hyperledger Besu*. Accessed: Nov. 12, 2019. [Online]. Available: https://www.hyperledger.org/projects/besu

[21] *Announcing Hyperledger Besu*. Accessed: Nov. 19, 2019. [Online]. Available: https://www.hyperledger.org/blog/2019/08/29/announcing-hyperledger-besu

[22] *Ethereum Blockchain Solutions*. Accessed: Nov. 19, 2019. [Online]. Available: https://pegasys.tech/

[23] *Remix*. Accessed: Nov. 19, 2019. [Online]. Available: https://remix.ethereum.org/

[24] *Smartcheck*. Accessed: Nov. 4, 2019. [Online]. Available: https://tool.smartdec.net/

[25] *Smartcheck*. Accessed: Jan. 26, 2020. [Online]. Available: https://github.com/smartdec/smartcheck

[26] *Eth Gas Station*. Accessed: Nov. 11, 2018. [Online]. Available: https://ethgasstation.info/

[27] G. Wood, "Ethereum: A secure decentralised generalised transaction ledger," *Ethereum Project Yellow Paper*, vol. 151, pp. 1–32, Apr. 2014.

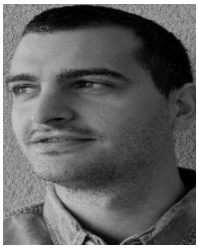
HAYA R. HASAN received the B.S. degree in computer engineering from the American University of Sharjah, UAE, in 2014, and the master's degree in electrical and computer engineering from Khalifa University, UAE, in 2018. She is currently a Research Associate with the Department of Industrial and Systems Engineering, Khalifa University. She has publications in her areas of interest, blockchain, as well as in security. She is passionate about research, especially in the field of blockchain and smart contracts.

KHALED SALAH received the B.S. degree in computer engineering, with a minor in computer science, from Iowa State University, Ames, IA, USA, in 1990, the M.S. degree in computer systems engineering from the Illinois Institute of Technology, Chicago, IL, USA, in 1994, and the Ph.D. degree in computer science from the Illinois Institute of Technology, in 2000. He is currently a Full Professor with the Department of Electrical and Computer Engineering, Khalifa University, UAE. He has over 220 publications and three US patents, and has been giving a number of international keynote speeches, invited talks, tutorials, and research seminars on the subjects of blockchain, IoT, fog and cloud computing, and cybersecurity. He is currently leading a number of projects on how to leverage blockchain for health care, 5G networks, combating deepfake videos, supply chain management, and AI. He has served as the Chair of the Track Chair of the IEEE Globecom 2018 on Cloud Computing. He is currently an Associate Editor of the IEEE Blockchain Tech Briefs, and a member of the IEEE Blockchain Education Committee.

RAJA JAYARAMAN received the Ph.D. degree in industrial engineering from Texas Tech University, the M.Sc. degree in industrial engineering from New Mexico State University, and the master's and bachelor's degree in mathematics from India. He is currently an Associate Professor with the Department of Industrial and Systems Engineering, Khalifa University, Abu Dhabi, UAE. His expertise is in multicriteria optimization techniques applied to diverse applications, including supply chain and logistics, healthcare, energy, environment, and sustainability. His research interests are primarily focused on using blockchain technology, systems engineering and process optimization techniques to characterize, model and analyze complex systems with applications to supply chains, maintenance operations planning, and healthcare delivery. His postdoctoral research was centred on technology adoption and implementation of innovative practices in the healthcare supply chains and service delivery. He has led several successful research projects and pilot implementations in the area of supply chain data standards adoption in the US healthcare system. His research has appeared in top-rated journals, including the *Annals of Operations Research*, the *IISE Transactions, Energy Policy, Applied Energy, Knowledge-Based Systems*, IEEE Access, the *Journal of Theoretical Biology*, the *Engineering Management Journal*, and others.

MOHAMMED OMAR was an Associate Professor and a Graduate Coordinator with Clemson University, Clemson, SC, USA. He was a part of the Founding Faculty Cohort of Clemson University Research Park, Greenville, SC, USA. He is currently a Full Professor and the Founding Chair of the Department of Engineering Systems and Management (currently renamed Industrial and Systems Engineering). His professional career includes a postdoctoral service at the Center for Robotics and Manufacturing Systems CRMS, and a Visiting Scholar at the Toyota Instrumentation and Engineering Division, Toyota Motor Company, Japan. He has over 100 publications in the areas of product lifecycle management, knowledge-based manufacturing, and automated testing systems, in addition to authoring several books and book chapters. He holds four U.S. and international patents. He was named a Tennessee Valley Authority Fellow of two consecutive years during the Ph.D. degree, in addition to being a Toyota Manufacturing Fellow. His group graduated seven Ph.D. dissertations and over 35 M.Sc. theses, and four Ph.D. students are currently on academic ranks in U.S. universities. His work has been recognized by the U.S. Society of manufacturing engineers SME through the Richard L. Kegg Award. He has also received the SAE Foundation Award for Manufacturing Leadership. In addition, he has received the Murray Stokely Award from the College of Engineering, Clemson University. He has also led an NSF I/UCRC Center and a part of the DoE GATE Center of Excellence in Sustainable Mobility Systems. His current research interests include capabilities in composite fabrication and manufacturing analytics at a laboratory Masdar City Campus. His current research group supported two Postdoctoral Scholar's Career Planning to become an Assistant Professor at the Texas A&M (TAMUQ), in 2013, and the University of Sharjah, in 2015. He currently serves as an Editor-in-Chief for the *Journal of Material Science Research* (Part of the Canadian Research Center), and as an Associate Editor for the *Journal of Soft Computing* (Springer), handling the areas of decision science, knowledge-based systems, in addition to his membership on several editorial boards and conference organizations. Furthermore, he serves on the Advisory Board of the Strata PJSC (part of Mubadala Aerospace).

IBRAR YAQOOB received the Ph.D. degree in computer science from the University of Malaya, Malaysia, in 2017. He worked as a Research Professor at the Department of Computer Science and Engineering, Kyung Hee University, South Korea, where he completed his Postdoctoral Fellowship under the prestigious Grant of Brain Korea 21st Century Plus. He worked as a Researcher and Deve-loper at the Centre for Mobile Cloud Computing Research (C4MCCR), University of Malaya. He is currently working with the Department of Electrical Engineering and Computer Science, Khalifa University, UAE. His numerous research articles are very famous and among the most downloaded in top journals. He has been listed among top researchers by Thomson Reuters (Web of Science) based on the number of citations earned in the last three years in six categories of computer science. He has been involved in a number of conferences and workshops in various capacities. His research interests include big data, blockchain, edge computing, mobile cloud computing, the Internet of Things, healthcare, and computer networks. He is currently serving/served as a guest/associate editor for various journals.



SAŠA PESIC is currently pursuing the Ph.D. degree with the Department of Mathematics and Informatics, Faculty of Science, University of Novi Sad, Serbia. He works as a Blockchain Engineer at VizLore Labs Foundation, Novi Sad, AZ, USA, and at VizLore LLC, Scottsdale, AZ, USA. He is currently a Visiting Researcher at the School of Computing, Informatics, and Decision Systems Engineering, Arizona State University, and at the Blockchain Research Lab, Tempe, AZ, USA. In his research work, he deals with the highly distributed Internet of Things and edge computing systems, analyzing their robustness, security, operating capacity, and stability. In addition, as a part of his doctoral thesis, he is designing, modeling, and implementing an advanced indoor positioning system called BLEMAT. He is the author/coauthor of nine conference papers and one paper in an international journal. As a Research and Development Engineer, he is actively working on two Horizon2020 Research Projects: PhasmaFOOD, Interconnect. In the past two years, he has worked on Vicinity, AgileIoT, and i SymbIote. His research interests include distributed ledger technologies and their interdisciplinary application in the domains of energy, finance, the security of the IoT systems, and peer-to-peer insurance.

TODD TAYLOR worked for Compaq, HP, and IBM for over 25 years in multiple technology and operations functions. He also co-founded Ops Rules, which is now a part of Accenture Analytics. He is currently a Professor of practice in the Supply Chain Department of Arizona State University's W.P. Carey School of Business. He is also the Founder of the ASU Blockchain Research Lab and the Co-Founder of Aperio–a blockchain solution provider and incubator which is now a part of Sweetbridge Inc. He is an expert in supply chain strategy and technology.

DRAGAN BOSCOVIC received the Ph.D. degree in EE and CS, numerical electromagnetic modeling from the University of Bath, U.K., in 1991. He is currently a Research Professor with the School of Computing, Informatics and Decision Systems Engineering (CIDSE), and a Distinguished Visiting Scholar at mediaX, Stanford University. He is also the Academic Research Director of AZ Applied Research Center, where his team's mission is to advance the research and development of blockchain-based technologies for use in business, finance, economics, mathematics, computer science, and all other areas of potential impact. He has 25 years of high tech experience acquired in an international set up, i.e., U.K., France, China, USA, and is uniquely positioned to help data-driven technical advances within today's global data-intensive technology arena.