Received November 29, 2019, accepted December 14, 2019, date of publication December 25, 2019, date of current version January 6, 2020.

*Digital Object Identifier 10.1109/ACCESS.2019.2962232*

# Cloud Manufacturing Architecture Based on Public Blockchain Technology

**BARAN KAYNAK[1], SÜMEYYE KAYNAK[2], AND ÖZER UYGUN[3]**
[1]Computer Research and Application Center, Sakarya University, 54050 Sakarya, Turkey
[2]Department of Computer Engineering, Sakarya University, 54050 Sakarya, Turkey
[3]Department of Industrial Engineering, Sakarya University, 54050 Sakarya, Turkey

Corresponding author: Baran Kaynak (kaynak@sakarya.edu.tr)

**ABSTRACT** With Industry 4.0, IT infrastructure has started to be used more effectively in the manufacturing sector. Cyber physical systems, IoT, cloud manufacturing, big data are some of the technologies that make up the concept of Industry 4.0. These technologies have solved many problems in the manufacturing sector. One of these technologies, cloud manufacturing technology, has emerged with the idea of pay as you go. This technology has enabled manufacturing resources to be leased and shared on a global scale. However, it has problems arising from its central structure and the need for a reliable 3rd party. Reliability, security, continuity, scalability, data lock-in, single point failure, data manipulation are some of the main problems. Blockchain (BC) is a decentralized and distributed technology. The data stored on the BC network cannot be altered in any way. With these features, we believe that BC-supported cloud manufacturing systems can overcome the aforementioned problems and eliminates the need for a reliable 3rd party. Based on this belief, in this study the agreements and communication between the resource provider and the customer, which is one of the basic functions of cloud manufacturing platforms, are realized with a decentralized application using BC-based smart contracts (SCs). The designed application is called the decentralized cloud manufacturing application (DCMApp). DCMApp does not operate on a fully public BC network, it has a hybrid structure and uses the Ethereum network as a public BC network. These features make DCMApp different from other BC-based cloud manufacturing applications. DCMApp's hybrid structure has enabled more transparent, economic and safe manufacturing agreements. It is also possible to store agreements on the BC network at a low cost without installing any server infrastructure. The use of Ethereum network makes it almost impossible to manipulate agreements.

**INDEX TERMS** Blockchain, cloud manufacturing, decentralized manufacturing.

## I. INTRODUCTION

Technological developments in manufacturing systems have changed the competitive factors in the sector. Before 1970s, cost, which is one of these factors, one the most important factor, whereas quality gained importance after the 70s. By 1990s, service and environmental factors gained importance and in the 21st century, information became the most important factor [1]. The information is the foundation of the 4th industrial revolution, namely, Industry 4.0.

The term Industry 4.0 collectively refers to a wide range of current concepts, including cyber-physical systems (CPS), internet of things (IoT), simulation, cloud computing, big data and advanced analysis techniques, service-oriented technologies, virtualisation and so on [2]–[8]. These technologies and paradigms provide solutions to different needs of manufacturing systems.

Combining recently emerged technologies with advanced manufacturing models and information technologies, Cloud Manufacturing is a new manufacturing paradigm that meets the needs of manufacturing systems. A number of significant researches have been made to develop cloud manufacturing technologies, design the system architectures, and define cloud manufacturing and key characteristics [9].

Cloud manufacturing is based on cloud computing. Cloud computing highlights two features: ease of access and shared use of resources [10]. Cloud manufacturing, with these two feature of cloud computing, increases customers' accessibility to manufacturing resources and capabilities through the Internet [11].

The associate editor coordinating the review of this manuscript and approving it for publication was Fabrizio Messina.

There is no standart definition of cloud manufacturing yet [12]. The first definition of cloud manufacturing was proposed by Li *et al.* [13]. Academic and industrial communities have shown great interest in this new concept. Many definitions of cloud manufacturing have been made in academic communities that highlighted different characteristics of cloud manufacturing.

Zhang *et al.* describe cloud manufacturing as service-oriented, knowledge-based, an intelligent network agile manufacturing model and technology [14]. Wang *et al.* describe cloud manufacturing as an integrated cyber physical system that provides manufacturing services on demand [15]. Adamson *et al.* describe it as a product realization model that enables rapid product development and innovation at minimum cost between cloud manufacturing service providers and customers [12]. Ren *et al.* describe it that "cloud manufacturing is a smart networked manufacturing model that embraces cloud computing, aiming at meeting growing demands for higher product individualisation, broader global cooperation, knowledge-intensive innovation and increased market-response agility" [16].

When the definitions of cloud manufacturing in the literature are examined, it can be said that cloud manufacturing is a new manufacturing paradigm that includes cloud computing, IoT, networked manufacturing model, cyber physical systems, service-oriented manufacturing model, virtual manufacturing model. Cloud manufacturing stands out with the following features:

- There is a shareable pool that holds a variety of manufacturing resources and capabilities [17].
- Manufacturer can acquire resources from service providers as per needs [18] and pay as you go.
- Systems at different points can communicate with each other via the Internet network and benefit from each other.
- Parts of cloud manufacturing applications can offer distributed service on different platforms.
- Different service users can run different platforms concurrently on the same infrastructure [19].
- Virtualization
- On-demand self-service.

These properties have eliminated many limitations of conventional manufacturing. In conventional manufacturing systems, the limits of the system are determined by the components that make up the system. There are limits to the range of parts or products that can be made on the system. The options of a company are only limited by partners in its network. Business partners and the company's resources around the system emerge as the biggest obstacle to the development of the system. Especially for SMEs, development will be limited to the budget as this will increase costs. During the manufacturing stage of a product, various resources and business partners are needed. In traditional manufacturing models, the procurement, maintenance, operation and all other stages of these partners or resources belong to the company. In order to overcome these problems caused by traditional manufacturing methods, managing each part with a cloud network by distributing responsibilities will improve productivity.

With cloud manufacturing, businesses can meet their needs without any investment by using a centralized and remotely accessible system. To take advantage of CM, users can use cloud manufacturing software interfaces. Cloud manufacturing platforms include 2 types of users: Service provider and customer. The service providers identify and market their advantaged resources and competitive capabilities in the cloud platform. Customers publish their customized demands, such as design drawings, process requirements, quality metrics, time of delivery and green standards [20] and discover and rent resources or functions available in the system. Service providers are the parties that take full responsibility for the services they offer in cloud manufacturing. The cloud user is offered the entire cloud service. Users only use the services they need. They don't have to own any resources.

Although the concept of cloud manufacturing provides solutions to many problems of traditional manufacturing, it remains insufficient at some points and causes some problems in the manufacturing. Some of the problematic points of cloud manufacturing can be highlighted below:

- Central systems need a reliable intermediary.
- In centralized or decentralized systems, resources can be rented to more than one person. But this situation causes data from different users to be stored at a single point in centralized systems.
- User data is stored and managed in the format designed by the application owner.
- Users can access their own data to the extent permitted by the cloud application.
- The lack of a standard spoken language among cloud platforms, keeping data in different forms in the application makes data access difficult and forms the basis for data lock-in problem.
- Responsibilities are distributed on cloud manufacturing platforms. You have to trust the companies that take responsibility. It creates a dependency in system [21].
- Users have limited control over the cloud application [22].

These features of cloud manufacturing make it inefficient in scalability, interoperability, reliability, security, data confidentiality, efficiency, continuity and flexibility [10], [23], [24]

It is possible to use blockchain (BC) technology to solve many of these problems. BC technology offers an innovative approach to decentralized and fully distributed mechanism in industries and businesses [25]. The BC technology basically can be considered as an immutable distributed ledger. BC is a technology that comes to the fore with guaranteeing the reliability of the data it stores. The BC technology has the following powerful characteristic:

- BC operates on a decentralized network. The system does not belong to a single entity. The decentralized structure of the BC technology increases the scalability of the network.

- Validation of the data in the network is provided by the whole network by consensus algorithms.
- After confirming each activity on the network, it is permanently recorded as a block. The blocks are linked to the previous block with hash information. So, transactions can be traced from the starting point to the final state.
- A copy of the approved information is available on all nodes on the network. This can be seen as a security vulnerability. But, this is a requirement to ensure reliability.
- Network management is provided by the rules of the network without a centralized management system.

BC inherently provides several important technological advantages such as durability, invariance, process integrity and transparency to users that are implications of its structural architecture [25]. Smart contracts (SC) has led to the development of many applications based on the BC network. There are studies that BC-supported CM applications can solve some problems of CM as in Li *et al.* [26].

Li *et al.* have designed BC-supported cloud manufacturing platform [26]. This platform operates in a peer to peer distributed network. The proposed platform is called ''BCmfg''. BCmfg platform is designed on a BC network installed using multichain software on virtualized servers. The BC network used in this study is designed for this model only, not a public BC network used for other purposes. Data on this BC network is encrypted and all data is recorded simultaneously on both the network and the database.

In this study, it is explained how to model and develop a BC-supported cloud manufacturing application using public BC networks. The developed application uses the public Ethereum BC network. The Ethereum is a network that supports SCs and offers a wide range of applications. The developed BC-supported cloud manufacturing application is hybrid; the data is not only kept on the BC network, but also some of the data is stored in the local databases on the users' computers. Important data is stored in the Ethereum network for a reasonable fee. To design a completely public cloud manufacturing application, all data have to be stored in the public BC network. But it is not possible to design such a system. Because the data size will be too large and too many write operations will have to be performed, it will increase the fees paid significantly. In this study a hybrid structure is preferred both for a decentralized application and to keep costs at a reasonable level. Storing important data on the Ethereum network makes the application run forever without the need for any central server infrastructure. The highlights of our preference for Ethereum network are as follows.

- The use of the public BC network allows the application to be used by any person.
- Ethereum has a market value and is the most popular network supporting SC. With Ethereum cryptocurrency, users can securely transfer money.
- The use of a known cryptocurrency network allows users to easily buy and sell this currency from the market. An unknown money transfer in an unknown network will make the system unsafe.

In this study a hybrid BC based decentralized cloud manufacturing application model is designed. In this model, it is ensured that the two parties can make a manufacturing agreement between each other and follow the cloud manufacturing process without the need of a third party by the help of SCs in BC network. The novelty of this study is that a decentralized CM application can be run on public BC networks using SCs with the help of hybrid model. The hybrid model has enabled more transparent, economic and trusty manufacturing agreements and communication. It is also possible to store agreements on the BC network at a low cost without installing any server infrastructure. In this model, communication, data transfer and fee transfer between the parties on a BC-based public network is presented. Also a decentralized application has been realized and it is explained how it can be developed with SCs in Ethereum network from public BC networks.

## II. RELATED WORK

### A. BLOCKCHAIN TECHNOLOGY

BC technology can be viewed of as a distributed database that holds a list of data records, or as a general ledger, keeping all processes shared and run between participants [27]. Unlike the classical database and ledger, it is decentralized and distributed.

A paper, ''Bitcoin: A Peer-To-Peer Electronic Cash System'', was published in 2008 by a person / group under the name Satoshi Nakamoto [28]. This article introduces electronic money that allows direct online payments from one party to another without an intermediary [29]. After a few months, in 2009, Bitcoin application was implemented [27]. Similar applications were developed using the basic features of Bitcoin cryptocurrency and the applications were labeled with the term cryptocurrency. Bitcoin, one of the most used cryptocurrencies, achieved a great success in 2019 with a capital market of approximately 223 billion dollars [30].

BC technology, which forms the basis of Bitcoin, became popular with the success of Bitcoin. Bitcoin has increased people's interest in BC technology, but it has also made people skeptical about this technology. Many economists have criticized Bitcoin. Many experts have advocated Bitcoin as the best investment tool as a global unit.

Bitcoin is based on classic BC technology. Classical BC technology suffers from data synchronization, double spending problems in distributed systems. The data synchronization problem has been attempted to be solved by applying the consensus model. The double spending problem is solved by a decentralized payment system rely on proof of work (PoW) [31], [32]. This solution was first applied by Satoshi Nakamota in the Bitcoin white paper [33]. With the improvements made, the BC technology currently used is slightly different from conventional BC technology. The BC technology used in this article is a completely distributed and decentralized system with P2P communication

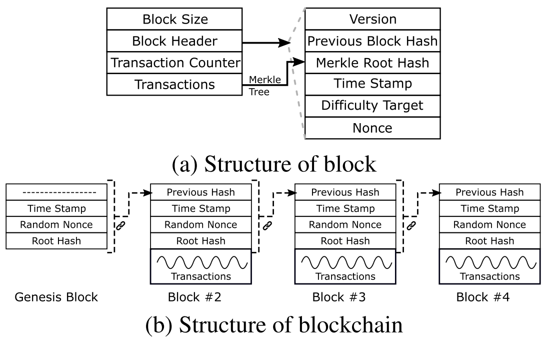


**FIGURE 1. Structure of blockchain.**

based on consensus and PoW models [33]. The general architecture of BC technology used in the study is shown in figure 1.

As shown in Figure 1a, each block has block size, block header, transaction counter and transaction fields. The block header field has 6 fields in itself: version, previous block hash, merkle root hash, time stamp, difficulty target and nonce. Version field allows you to monitor the software protocol. The previous block hash field holds the hash value of the previous block. If there is any change in the transaction field, the value in the hash field changes. This area ensures data security. The Merkle root field holds the hash value of the merkle tree in BC. Approved transactions are included in the merkle tree, and if there is any change in the transaction, the root hash value of the local tree associated with the transaction changes, and all blocks created after the modified block are lost. The Difficulty target field indicates the degree of difficulty of the PoW algorithm. The nonce value is a random value used for the proof of work algorithm.

As shown in Figure 1b, the BC network starts with the genesis block. The genesis block has different areas than the other blocks and represents the starting block of the chain. The Genesis block is created by the person who started the chain. This block contains the basic structure and rules of the BC.

Users must register to the network to take action on the network. The registered user has public and private keys. Any person with private key is authorized to perform all transactions for this account. The transactions formed by the people in the BC are evaluated by consensus algorithms. Transactions validated by consensus algorithm are included in the chain as a block. The blocks are seen by all people in the chain.

Bitcoin is based on classic BC technology and was developed for value transfer only. It is not suitable for the development of applications in different areas. The development of BC-based applications in different areas is made possible by SCs. SCs contain executable functions and status variables [34]. When developers design and submit a SC (SC) to the network, the logic of the SC's content can never be changed by anyone. People who accept and approve the SC are obliged to apply the contents of the SC. Execution of the code under certain conditions in the BC network is guaranteed by SCs [34]. With the unique features and development of BC technology, BC technology is used not only in value transfer but also in many areas such as gaming, supply chain, health, education and proof of ownership [35], [36].

Applications may need different security level, network access permission, speed level, network structure according to their purpose. Some applications need a completely decentralized network structure, while others may accept partial centralization. There are different types of BC technology to meet these needs. These types are public, private and consortium BC. In the public BC which is all decentralized, everyone on the network can access all of the records, and participate in the consensus process. This does not mean that the network is not secure [26]. Security is provided by cryptography. Bitcoin application is an example of public BC network. In the consortium BC network, the selected nodes participate in the consensus process. This leads to partial centralization. Access to records on the network can be restricted. In the private BC network, the owner of the BC network participate in the consensus process. Nodes approved by the person / company managing the network can join the private BC network [37].

The participation of the whole network in the consensus process in the public BC network makes it almost impossible to modify or delete transactions created in this network. In the consortium and private BC networks, transactions can be easily modified because they have a limited number of participants and only the allowed nodes are included in the consensus process. In these networks, security and reliability depend on the network's administrators. As everyone can join the public BC network, the network grows rapidly and the security of recordings is enhanced. The use of the public BC network is less costly than the creation of the consortium and private BC network. Any SC supported public BC application and architecture can be used cost-effectively in different applications. This allows for faster development of applications.

Ethereum is a public BC network application that supports SCs. In this study, Ethereum network is used to take the advantages of public BC network and SCs. The Ethereum network is the second largest BC network after the Bitcoin network. As malicious people take over the majority of the network, the information on the network can be changed and the wrong information confirmed. The large network will prevent this bad scenario. For this reason, using Ethereum network which is accepted all over the world with a large number of participants in the system will increase the reliability of the system.

## B. BLOCKCHAIN SUPPORTED DAPP

The data of recently popular applications such as Whatsapp, Twitter, etc. are stored on servers that are responsible for a single organization / person. Such applications are centralized applications. Centralized applications have limitations such as security, the need for reliable third-party, less transparency, single point failure. Because of these limitations, it may be necessary to avoid a central structure in all or some of the applications.

Decentralized applications (dApp) are distributed applications running on a peer-to-peer (P2P) network. Application data is not maintained on a single server, and multiple nodes on the network have copies of the data. Torrent applications are examples of dApps running on P2P networks. Large files are shared with the torrent application. There are many websites that provide access to these files and search engines. Thus, the loss of a node and the loss of access to the node do not prevent accessibility to the system. As the number of people sharing files increases, the file downloads faster. When the sharing stops, the network breaks and the previous steps and the shared file disappear.

DApps built on the basic architecture of BC technology have some different features from traditional dApps:

- The application should use a crypto token.
- The application must be open source.
- Application data must be stored on a decentralized BC.
- Consensus algorithms (PoW based algorithm, Proof of Stake, Delegated Proof of Stake algorithm) should be used.
- With SCs, local functions in dApp have become functions that run on the BC network. So, It is guaranteed that the functions cannot be changed.

Thanks to these features, BC-supported dApps are more reliable. Data cannot be modified or deleted in BC-supported dApps. In dApps like Torrent, history is not kept, data loss occurs, event flow can not be stored.

Figure 3 shows a basic flow of BC-supported dApp. A transaction is created. The transaction, referred to as a node, are notified to p2p network of computers. The transaction is verified by a consensus algorithm by a group of computers, by a single computer, or by the entire network according to the BC type and verified transaction connects to the next block. Thus, a new data block is added to the ledger. The transaction is now permanently stored in the BC network.

On the Ethereum platform, dApps are performed with SCs. For dApps created with SCs, registration is performed only with a special address without any personal information. The rules and core structure of the application are written using the solidity language supported by SCs. To develop a BC-supported dApp, it is possible to use any software language that can communicate with the BC network. In this study, application is developed by using C# language on dotnet core framework. The details of the application are described in Chapter IV.

## III. IMPORTANCE OF DAPP IN CLOUD MANUFACTURING

Cloud manufacturing applications/architecture structure is shaped according to past experiences, different perspectives of industry and academic environment and solutions offered to different needs. Despite the differences in cloud manufacturing applications, applications have common features. A cloud manufacturing application is centralized for whatever purpose it is prepared for. There are many problems that cloud manufacturing platforms have because of their central structure.

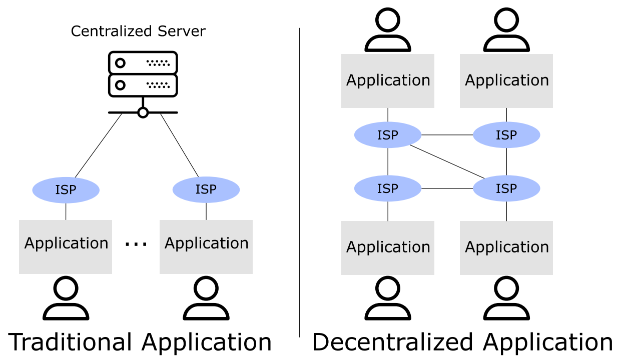


**FIGURE 2. Traditional application vs decentralized application.**

In order to implement cloud manufacturing platforms, the system needs to be designed and installed. Installation and maintenance of such systems are costly. This cost must be met in some way from users. Users must pay this cost to receive services from the application. Since cloud manufacturing platforms are applications running over the internet, it is not possible to benefit from the functions of the application by downloading the application to the computer once. However, in a dApp, users are not connected to any center. They can run the functions offered by the application from any point. If the functions offered by cloud manufacturing systems are provided to users within a BC supported dApp, users can use these functions freely with an application that is independent of a central point and can transfer value between the parties. In summary, BC-supported decentralized cloud manufacturing applications will eliminate the need for a central server, and users will not be charged for the operation and maintenance of the central application. As shown in Figure 2, a conventional application needs a central server, whereas decentralized applications do not need any central server.

In centralized application, access to the application takes place from a single physical point. The application may become inaccessible due to damage to the physical server, attacking the application, or terminating the application. This will interrupt the process and cause serious losses. Failure in cloud manufacturing systems results in material losses for both the service provider and the receiving party. In order to provide uninterrupted service, the platform must have a high quality network connection and the physical machines on which the application is running must be hassle free and secure [19]. In BC supported dApps, the continuity of service will not depend on a single point. In case of a problem in any node of the BC network, the continuity of the service can be easily achieved by connecting to another node. The BC-supported decentralized cloud manufacturing application can be run continuously from any point and provide the functionality needed between the parties.

The same application serves many users in cloud manufacturing systems. Within the application, user data is stored in a common database. The user logging in to the application, with the authorization given to him/her, makes transactions on the screens related to him in the application. A user authentication system is used to allow the user to log into the application. These systems are usually based on user name and password information. There are also applications that use 2FA method (two factor authentication) to increase the security of the login system. However, these security measures are designed to protect users from external threats only. It does not offer any security measures against the threats of those who manage or develop the application. Any person who has access to the application's database is a potential threat to the system. In the BC-supported cloud manufacturing application, the user stores the data that are verified by a large audience in the BC network. By running the application on its own computer, the user can do all the operations by using the secret key of his wallet without sharing the user name and password information with a remote server.

In central cloud manufacturing applications, data remains locked within the application if the application is inaccessible. The user cannot access his/her data in an emergency or extract it from the central application for use on another platform. Even worse, the application can completely delete this data after a certain period of time or may sell this data. For example; The online storage service called Linkup was closed in 2008 after it lost access to more than 45% of customer data [38]. Linkup relied on Nirvanix to store customer data. 20,000 Linkup users were no longer able to receive services. 20,000 Linkup users' data was lost. Keeping user data locked within the platform is a user guarantee for app owners, while leaving users vulnerable to increased prices, reliability and accessibility problems.

Increasing the number of users of the application is of great importance for the application owner. In a cloud manufacturing application, increasing the number of users increases product diversity and shopping. Shopping takes place in two ways. The purchaser sends the fee to the application or directly to the owner of the product. If the application acts as an intermediary during the shopping process, the fee is kept in the bank account of the application acting as an intermediary. The fee is sent to the person who sells the product with the approval of the person receiving the product. The intermediary application receives commission fees for all these transactions.

In cloud manufacturing applications, users need a reliable intermediary and the functioning of the process is controlled by the institution or persons, not by the system. The intermediary institution/persons may choose any party in a possible dispute between the parties and not be impartial. In the BC network, SCs guarantee agreements between the parties. The parties must fully comply with the rules contained in the SC. So, the security of the agreements will be guaranteed. Agreements with BC-supported dApp are verified by the entire network. The agreements are integrated into the processes of BC-supported dApp applications. No intermediary is needed. For example; If it is planned to make a fee transfer in case the relevant conditions are met, the fee in the system is not transferred until the terms of the agreement are fulfilled. There is no need for a bank or intermediary company for fee transfer. The security of the money is provided free of charge.

## IV. A DAPP APPLICATION FOR MANUFACTURING PROCESS MANAGEMENT IN THE CLOUD MANUFACTURING PLATFORM

In this study, the basic form of manufacturing cycle realized in cloud manufacturing environment is realized as decentralized and distributed with the help of SCs. This distributed application uses the Ethereum network with a public BC network. There are a number of requirements that this application must meet are listed below.

Requirements:

- Businesses or individuals must be able to register the machines (resources) they own.
- The customer must be able to register the job to the system.
- The customer must see resources that may be appropriate for the job.
- The customer must be able to send proposals to the resource provider for the work to be done.
- Resource provider must be able to see and evaluate job offers.
- All transactions must be operated on a SC.

As in cloud manufacturing, the audience who wants to get the job done in this application is called the customer, and the audience who does the job is called the resource provider. In this application, the resource provider is the owner of resources to be used in manufacturing. The resource provider must accurately define all the properties of their resources when uploading their resources to the system. In this application, customers are the people who upload their job descriptions to the system and search for the relevant resources in the system and ask for the most suitable resource to be done.

Users need a BC wallet to operate on any BC network. Since the Ethereum network is used in this application, users are required to create an Ethereum wallet. This can be easily done with various wallet applications for free. Wallet is a crypto-protected tool with an address and a private key to perform transactions on the network. The private key needs to be protected very well. Once the user has a wallet, they can start using the application. The sequence diagram of the application is shown in the Figure 3.

The resource provider registers its resources in the distributed cloud manufacturing application (DCMApp). During the registration process, the necessary information is collected through the interface and written into a SC. A unique SC address is created. This address will now be used for all operations on the resource. In order to perform this transaction, the resource owner has to make a transaction using his/her own wallet. It is mandatory for all users to make transactions with a wallet. It is not enough to only save resources to DCMApp. Users should also be able to view

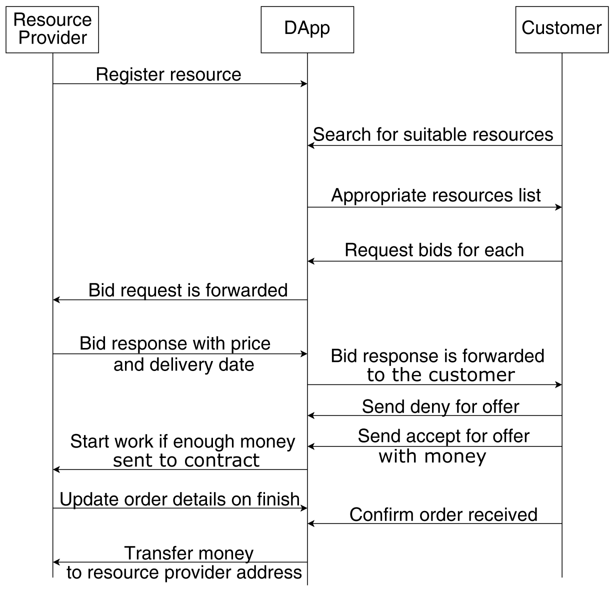

Resource Provider
DApp
Customer
Register resource
Search for suitable resources
Appropriate resources list
Request bids for each
Bid request is forwarded
Bid response with price and delivery date
Bid response is forwarded to the customer
Send deny for offer
Start work if enough money sent to contract
Send accept for offer with money
Update order details on finish
Confirm order received
Transfer money to resource provider address

**FIGURE 3. Application sequence diagram.**

saved resources. For this, the SC addresses created for the sources should be collected in a list. This list can be stored in another SC or in any database. If the list is stored in an SC, for each new item added, it must write to the SC where the list is held. This will increase the cost. However, the list will be able to be stored in a distributed manner. The distributed storage of the list ensures that the list cannot be changed unchecked. If the list is stored in a central database, the cost will be significantly reduced, but; the reliability problem will arise. In this study, it can be said that it would be more appropriate to store it in a database since manipulation of the list of sources addresses will not cause a serious security problem.

After selecting the relevant resources, the customer enters the details of the job and the deadline. The application creates a customer-specific SC in the BC network and all remaining operations continue through the SC. The customer makes quot through the application to the selected sources. The quote process initiates a workflow for the relevant resources within the SC. During the operation of the SC, the workflow specified in the SC is fully monitored. This process is pre-processed into the SC and propagates the entire network. So, changing the data in the SC is no longer possible.

When requesting a quote, it is written to the Job Contract (JC) created for the job. The quotation must be notified to the resource provider. This notification can be processed to the SC of the resource or sent to the contact address specified in the SC of the resource by email or similar methods. Sending the notification via email or SC will not make any changes to the operation of the process. Therefore, sending by email is seen as a logical solution because it will provide faster notification and cost less. Only the resource owner can respond to this quote within JC. The resource owner acquires information about the business after reviewing the quote. The resource owner creates a job plan for that resource and generates the quotation. Once the resource owner has made the preparations, he/she enters the quote and information on when to deliver it to JC. In this way, the resource owner has submitted his offer to the BC network through SCs.

The customer reviews the offers made by all source providers. The customer may accept one of the offers or reject all offers and request new offers. If the customer accepts, JC will now complete the entire offer process. The fee offered by the customer is recorded in the SC. Thus, both the customer and the source provider are secured. The customer pays the fee with the existing Ethereum in the wallet. After the customer makes the payment to JC and gives the job confirmation, JC's status now changes to "JobAssigned". When the resource owner completes the job, he sends the order to the customer and the document number obtained according to the delivery method is saved to JC. The job status is now saved to JC as "Sent" and a counter is started. This counter will be used for automatic confirmation mechanism.

After the customer receives the order, the confirmation method in JC is called. This method transfers the money in JC to the resource owner's wallet and marks the JC as "Completed". If the customer does not confirm receipt of the order, at the end of the maximum approval period, the source owner is entitled to receive the fee from JC. The maximum approval time is the information specified when creating JC. All transactions between the customer and the sourcing provider are recorded on the Ethereum network with a SC. SC data is permanently stored within the Ethereum BC network.

## A. SMART CONTRACTS

SCs are permanently registered to the system as an application instance when deployed to a BC network. A new application instance is created for each instance created from the same SC code. A SC code contains variables and functions. In this application, when using SCs, it is discussed whether to store all jobs in a single SC or to create a new SC instance from the same SC code for each job. Both options will make the system work. However, if we compare these two methods, it can be understood why it would be more appropriate to create a new SC instance for each job.

In case all operations are performed in a single SC instance;

- Advantages:
  - There is no charge for the new SC deployment.
- Disadvantages:
  - All operations are stored in the same SC.
  - It is difficult to update.
  - Since a single SC address will be generated, the use of the same address may cause security problems.
  - It is very difficult to maintain in the long-term.
  - The person deploying the SC may have access to the SC.

In case a new SC is created for each business agreement;

- Advantages:
  - Only one job's information will be stored in the SC.

- It is possible to use a more up-to-date SC code immediately in a new business agreement.
- Creating SC addresses separately for each job will ensure that only those involved know this SC address.
- Since the person requesting the job does the SC deployment, no unknown person will have stored any information on the SC.
- When the job is finished, all functions of the SC will be sealed.

- Disadvantages:
  - Re-deploying SC for each new business will increase the cost.

### B. IMPLEMENTATION OF DCMAPP

The solidity and C# programming languages are used in the implementation of DCMApp. Ethereum SCs are written using solidity programming language. The application is developed using C# language with dotnet core framework. Dotnet core framework has been chosen because it is open source and cross platform. In addition, it is preferred because developers have experience in C# programming language.

The application has a distributed structure thanks to its ability to connect to the Ethereum network. In order not to be a centralized application, more than one sample should be able to solve the same problem, regardless of a single point. However, preparing a completely distributed and completely independent application from a single point poses a number of challenges. For a completely independent application design, the following questions should be answered; "Where will the data be stored?", "How will the application be distributed to users?", "How will version management be performed?", "How to store common data?".

In this application, the data that must be stored in a distributed structure, is stored in SCs. In order to achieve a completely distributed structure, storing all information in a SC would be a very costly solution. It is therefore conceivable to store a set of data with a non-distributed solution. Storing data in a local database within the application, which will not compromise agreements, would be a cost-effective solution. The most important principle of the application is to never compromise reliability. SCs, which ensure the safety of the process, store and operate the key information of the process. However, during the operation of the process, a number of detailed information is kept in local databases by different methods. This detail information; a job design file, SC addresses of resources saved on the network, capabilities of these resources, and many more. Storing this information on a central server, can be seen as a solution. However, it would be undesirable for a distributed application to operate while adhering to a central point. Whether the application is connected to a central point of a person or institution, both in terms of security and continuity, will cause serious problems in the long term. Various cloud storage solutions can be a helpful tool at this point. This solution will be a very

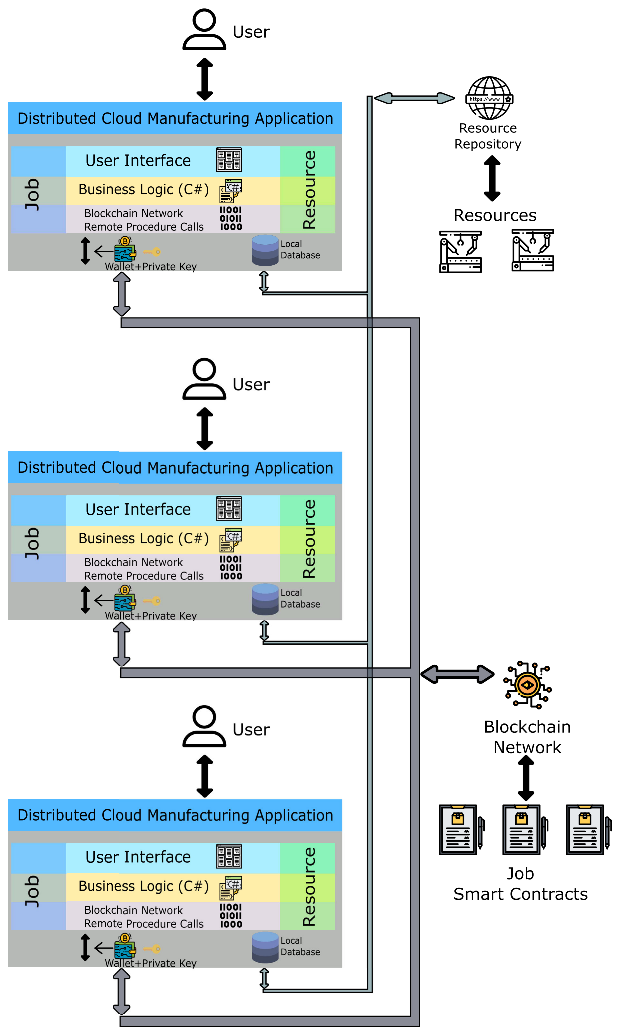


**FIGURE 4.** Application diagram.

practical solution especially for private sharing between the parties. As an alternative, the IPFS system, which can also run in a distributed manner, is used to store data on users. The data contained in the IPFS system is stored in a distributed manner throughout the world. The user can store and share his / her own data. In this study detail information data is stored in a central repository and syncronized with client's local database. This data also can be stored in IPFS or other cloud storage solutions in the future.

Since the application is prepared by using dotnet core, it can be packaged and run one copy on the users' computers. If users cannot do this, only one copy of the application can be run on one central or multiple servers to simplify operations. Thus, only knowing the web address will be enough for the application to be used. This does not prevent the application from being distributed. The operations will continue to be performed in the same way on the Ethereum network. The structure of the application is seen in Figure 4. The application communicates with Ethereum network with rest based web services. The NEthereum library is used in the Ethereum network to operate in C#. The NEthereum library allows you to perform operations on the Ethereum network using the C# language only. You can deploy a SC written in solidity language to the network with this library and execute its functions. SCs prepared with Solidity are converted to C# classes with NEthereum. These classes include bytecode and ABIs of SCs. Proxy methods are created automatically for all functions that SCs have.

There are two basic SCs within the application. The first SC is prepared for resources. It is called resource smart contract (RSC) The second is the SC code prepared for the jobs to be done in the sources. It is called job smart contract (JSC). The RSC variables can be seen in the code 1.

```
//The wallet address of the resource and the
    smart contract's owner.
address owner;
//Resource name specified by resource owner.
string name;
//Details of resource features and
    capabilities. Size, color, material etc.
string properties;
//The information whether the resource is
    active or accepting a job is kept.
bool isActive;
//The total number of jobs received. This
    number will give the customers an idea
    when choosing the source, such as how
    much the source is preferred, how
    reliable it is.
uint256 jobCount;
//Average customer rating for this resource
    after the job is completed.
uint256 customerPoint;
```

**Code 1. Resource smart contract variables.**

In the RSC, only the resource is defined. An RSC should be created for each of the resources that will serve the cloud manufacturing environment. However, the addresses of all these definitions should be collected at one point. This problem can be solved with another SC, where all resources have SC addresses, but will be costly and difficult to maintain. Instead, it would be much more practical and inexpensive to store only addresses on a central server and distribute them to all users. Using a remote resource repository, which is hosted on a central server, users will be able to sync data into their local databases.

JSC are redeployed to the network for each job. For each job, the customer uploading the job to the system must deploy a SC code through the application. This can be done easily through the application. Enumerations, structs, variables and functions are available in JSC design. Enumerations describe the status of the job and the offer. Enumeration JobStatus defines 6 different states of the job. These enumeration values are as follows: Created, OffersPending, JobAssigned, Sent, Confirmed, Deleted. OfferStatus enumeration defines OfferRequested and OfferResponded. The Offer struct, in which an offer is represented, is defined as code 2.

In JSC, defined variables can be seen in the code 3. Some of these variables are used in constructor method while JSC is created and some of them are used in other functions.

```
struct Offer{
  //Offerer's wallet address
  address offerer;
  //Date the offer request was created
  uint requestDate;
  //Date the offer was responded
  uint responseDate;
  //Status of the offer
  OfferStatus enumOfferStatus;
  //The date on which the offer undertakes to
      complete the job
  uint finishDate;
  //Price quotation for this job
  uint price;
  //Unique value generated for the offer
  uint id;
  //The resource address of the offer
  address resourceAddress;
  //Whether the offer has been accepted
  bool accept;
}
```

**Code 2. Offer struct in JSC.**

Basically, the JSC's date information, status, a list of offerers recorded, acceptance status and date information of the process after the offer is accepted, and information such as post-delivery information are stored in these variables. The user who creates the JSC always calls the constructor method during creation. Details of the constructor method can be seen in code 4. In addition, by default in the constructor method, the caller is assigned as the contract owner, the current date is assigned to the createDate variable, and the job status is assigned as Created.

After calling JSC constructor method, variables are initialized. JSC's functions are now ready to use on the BC network. These functions can be seen in code 6. All of these functions are operated by checking a number of conditions before being operated. For example, a job offer can only be made by the user using the contract owner address, or the response to an

```
JobStatus enumJobStatus;
address contractOwner;
uint createDate;
uint dueDate;

uint resourceTypeID;
string propertiesJson;
string designUrl;

address payable acceptedOfferer;
uint acceptDate;
uint confimationMaxTime;
uint confimationExtendTime;
uint confirmDate;

string trackingCode;
uint sentDate;

bytes1 constant maxExtendCount = 0x03;
bytes1 extendCount;

mapping(address => Offer)  public offers;
uint offerCounter;
```

**Code 3. JSC variables.**

```
constructor(
  uint _dueDate,
  uint _resourceTypeID,
  string memory _propertiesJson,
  string memory _designUrl,
  uint _confimationMaxTime,
  uint _confimationExtendTime
  ) public
{
  dueDate = _dueDate;
  resourceTypeID = _resourceTypeID;
  propertiesJson = _propertiesJson;
  designUrl = _designUrl;
  confimationMaxTime = _confimationMaxTime;
  confimationExtendTime =
      _confimationExtendTime;
  contractOwner = msg.sender;
  createDate = now;
  offerCounter = 0;
  enumJobStatus = JobStatus.Created;
}
```

**Code 4. JSC constructor.**

offer can only be made by the resource owner whose wallet address is registered in the bid list. This type of validation is performed with the "require(boolean)" function of the solidity language in all functions. Example responseOffer function is given in code 5. Users cannot operate these functions unless the necessary conditions are met.

```
function responseOffer(bool accept, uint price,
     uint finishDate) public returns (bool) {

  require(offers[msg.sender].id > 0);
  require(enumJobStatus == JobStatus.
      OffersPending);

  offers[msg.sender].accept = accept;
  offers[msg.sender].price = price;
  offers[msg.sender].finishDate = finishDate;
  offers[msg.sender].responseDate = now;
  offers[msg.sender].enumOfferStatus =
      OfferStatus.OfferResponded;
  return true;
}
```

**Code 5. JSC responseOffer function.**

Each function defined in the SC will be executed on a computer on the Ethereum network. Any changes to the SC variables will be processed into blocks in the chain and distributed to the entire Ethereum network. Therefore, the operations performed on SC are trusted. Any client that is part of the Ethereum network will be running web services that accept remote calls. SC functions can be operated by remote calling to any computer in the Ethereum network. This makes it possible to integrate SCs with other systems.

The site map of the application is given in Figure 5. The application has two main menus, Resources and Job. Within the Resources menu, menus are designed in which resource providers can manage their resources. Under the Jobs menu, there are screens that can create and manage business agreements for a job. Job adding / editing screen can be seen in Figure 6. Customer see him job list from job list screen

```
function requestOffer(address offerer, address
    resourceAddress)
                public returns (uint)
function responseOffer(bool accept, uint price,
     uint finishDate)
                public returns (bool)
function acceptOffer(address payable offerer)
    payable
                public returns (bool)
function deleteJob() payable
                public returns (bool)
function sendFinishedJob(string memory
    _trackingCode) payable
                public returns (bool)
function confirm() public returns (bool)
function requestConfirmTimeExtend()
                public returns (bytes1, uint)
function offerCount()
                public view returns (uint)
function withdraw()
                public returns (bool)
```

**Code 6. JSC functions.**

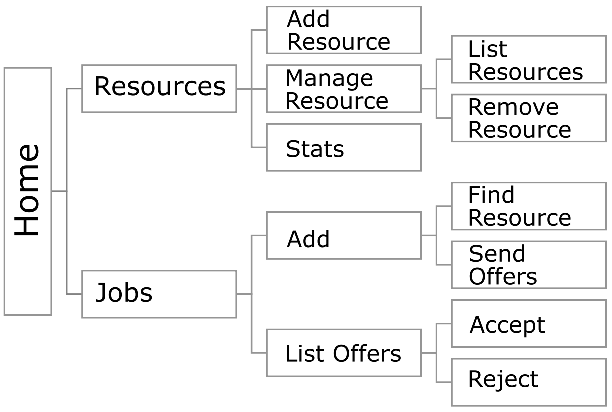


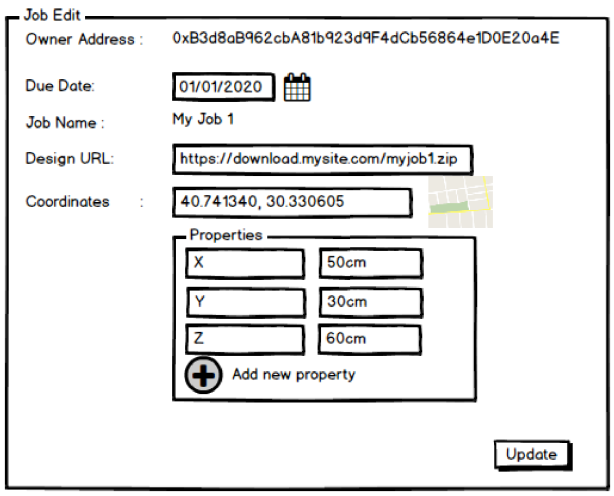
**FIGURE 5. Application sitemap.**

which is shown in Figure 7. From the job operations screen (Figure 10) users can go to resources and list offers screens. In order to find resources resource finder screen (Figure 9) lists available resources from the resource repository. Figure 8 shows offers status and also shows contract details of selected offer. Within the Resources menu, menus are designed in which resource providers can manage their resources. Resource owners can manage their resources into the resource repository. Related screens can be seen from Figure 11.

## V. RESULTS AND DISCUSSION

The proposed model is evaluated under two headings in terms of applicability and reliability. In terms of applicability, there are some limitations on how the model can be configured on a public BC network, unlike other distributed cloud manufacturing applications. In terms of reliability, evaluations about how the elimination of intermediaries contributed and how it can provide a trust environment were expressed.

An example study has been conducted on how cloud manufacturing systems can be operated on a public BC

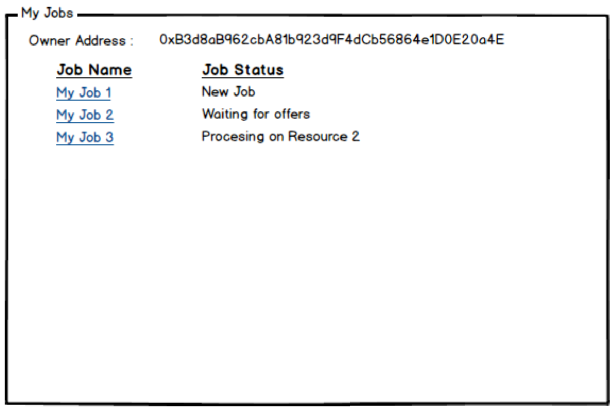

Job Edit

| | |
|---|---|
| Owner Address : | 0xB3d8aB962cbA81b923d9F4dCb56864e1D0E20a4E |
| Due Date: | 01/01/2020 |
| Job Name : | My Job 1 |
| Design URL: | https://download.mysite.com/myjob1.zip |
| Coordinates : | 40.741340, 30.330605 |

Properties

| | |
|---|---|
| X | 50cm |
| Y | 30cm |
| Z | 60cm |

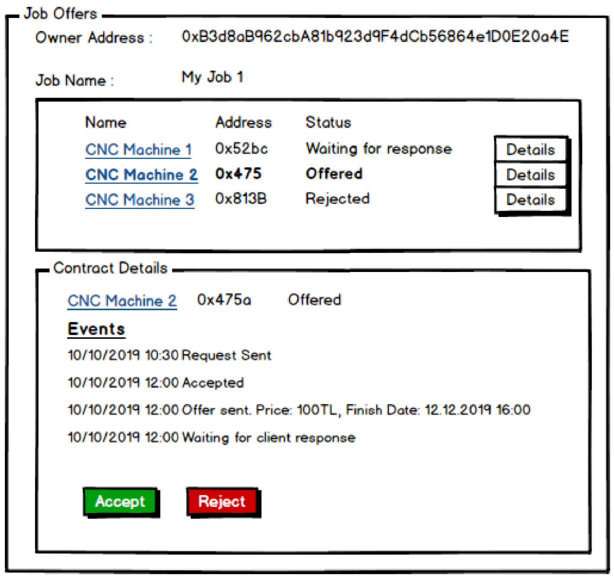
FIGURE 6. Job add/edit screen.


My Jobs

Owner Address : 0xB3d8aB962cbA81b923d9F4dCb56864e1D0E20a4E

| Job Name | Job Status |
|---|---|
| My Job 1 | New Job |
| My Job 2 | Waiting for offers |
| My Job 3 | Procesing on Resource 2 |

FIGURE 7. Job list screen.


Job Offers

Owner Address : 0xB3d8aB962cbA81b923d9F4dCb56864e1D0E20a4E

Job Name : My Job 1

| Name | Address | Status | |
|---|---|---|---|
| CNC Machine 1 | 0x52bc | Waiting for response | Details |
| **CNC Machine 2** | **0x475** | **Offered** | Details |
| CNC Machine 3 | 0x813B | Rejected | Details |

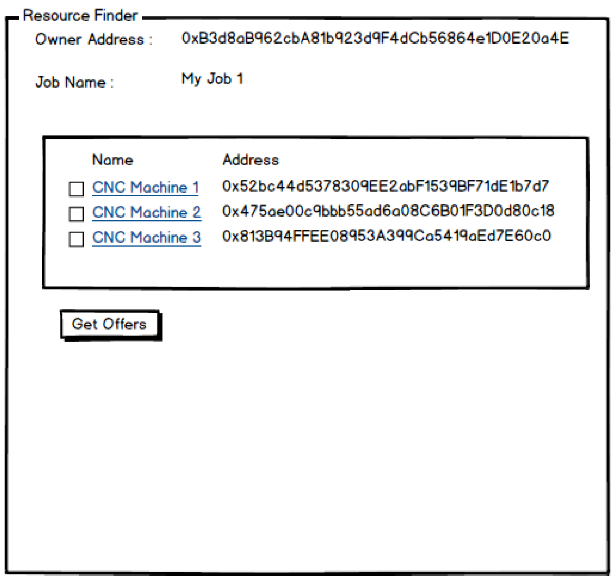
Contract Details

CNC Machine 2 0x475a Offered

**Events**

10/10/2019 10:30 Request Sent
10/10/2019 12:00 Accepted
10/10/2019 12:00 Offer sent. Price: 100TL, Finish Date: 12.12.2019 16:00
10/10/2019 12:00 Waiting for client response

FIGURE 8. Job offers screen.

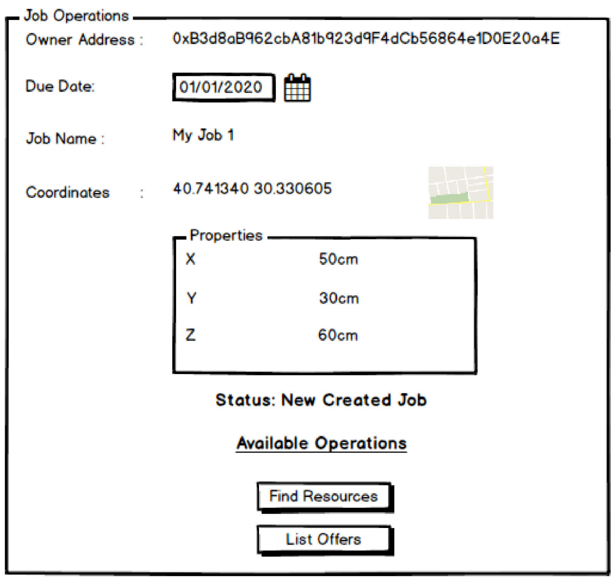

Resource Finder

Owner Address : 0xB3d8aB962cbA81b923d9F4dCb56864e1D0E20a4E

Job Name : My Job 1

| Name | Address |
|---|---|
| ☐ CNC Machine 1 | 0x52bc44d5378309EE2abF1539BF71dE1b7d7 |
| ☐ CNC Machine 2 | 0x475ae00c9bbb55ad6a08C6B01F3D0d80c18 |
| ☐ CNC Machine 3 | 0x813B94FFEE08953A399Ca5419aEd7E60c0 |

FIGURE 9. Job resource finder screen.


Job Operations

| | |
|---|---|
| Owner Address : | 0xB3d8aB962cbA81b923d9F4dCb56864e1D0E20a4E |
| Due Date: | 01/01/2020 |
| Job Name : | My Job 1 |
| Coordinates : | 40.741340 30.330605 |

Properties

| | |
|---|---|
| X | 50cm |
| Y | 30cm |
| Z | 60cm |

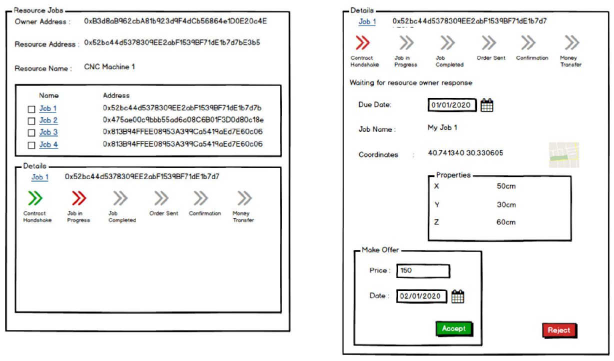
**Status: New Created Job**

**Available Operations**

FIGURE 10. Job operations screen.

network. With the developed application, users can share job-resources in a distributed manner without a central structure on a cloud manufacturing system. The difference of this application from traditional cloud manufacturing applications is that agreements are made on BC.

The use of a public network has enabled the application to be used by any person without any permission or membership. The data privacy provided by private BC can be ensured by encryption methods in public BC networks. Transaction confidentiality provided by the private BC network is not possible in public BC networks. This feature makes the transactions performed in the application more transparent. This application is designed to work on the public BC network. However, some necessary information is not stored in the public BC network, rather it is stored in the local database. This database receives resource information from another repository outside the BC network. This repository can be a server in a centralized or distributed structure. What is


Resource Jobs
Owner Address : 0xB3d8aB962cbA81b923d9F4dCb56864e1D0E20a4E
Resource Address : 0x52bc44d5378309EE2abF1539BF71dE1b7d7bE3b5
Resource Name : CNC Machine 1

| Name | Address |
|---|---|
| Job 1 | 0x52bc44d5378309EE2abF1539BF71dE1b7d7b |
| Job 2 | 0x475ae00c9bbb55ad6a08C6B01F3D0d80c18e |
| Job 3 | 0x813B94FFEE08953A399Ca5419aEd7E60c06 |
| Job 4 | 0x813B94FFEE08953A399Ca5419aEd7E60c06 |

Details
Job 1 0x52bc44d5378309EE2abF1539BF71dE1b7d7

Details
Job 1 0x52bc44d5378309EE2abF1539BF71dE1b7d7
Waiting for resource owner response
Due Date: 01/01/2020
Job Name : My Job 1
Coordinates : 40.741340 30.330605
Properties
X 50cm
Y 30cm
Z 60cm
Make Offer
Price : 150
Date : 02/01/2020

**FIGURE 11.** Resource jobs screen.

important here is not to store all information in a public BC network that requires a processing fee. System should be able to transfer detailed information such as product design, resource information, special messages through different channels. These channels can be storage areas on private servers or cloud services. The information transferred outside the BC network should not violate the agreement between the parties. The size of the information is very important since there is a transaction fee in the Ethereum network. The information on the Ethereum network should only contain the basic information of the agreement between the parties.

By choosing public BC network, the existing BC network is used and thus no additional cost is required. In this way users are able to use the existing BC network without any investment. Thus, users will only make one payment per write transaction using the public BC network. In the private BC or consortium BC networks, there are administrator(s). These networks are managed by specific person(s). An operating cost must be incurred in order for the network to survive continuously. However, in this application, there is no additional charge for the survival of the network.

With DCMApp, users can take part in the system directly through public BC networks without being a member of a group. However, in private and consortium network models, a network have to be created. In order to be a part of this network, permission must be obtained from the administrators who created network, and perhaps a fee may be requested. From an individual perspective, the proposed model is more economical than the private and consortium networks for the reasons mentioned above. The total cost spent on the network varies according to the purpose and scale of the application. In a cloud manufacturing application that is intended to be realized in consortium and private networks, the total cost spent on the network may be less than the public BC depending on its scale. From an individual perspective, it can be said that the use of public BC networks is more economical.

SCs enable users to make bilateral agreements on the BC network. These agreements with SC inherit the security features of the BC network. Thus, additional security measures are not required to secure the agreements. In this study, it is shown how to operate a manufacturing application, not only transferring value on the BC network using SC. The Ethereum network that supports the public BC network and supports SC has been used in this study. The Ethereum network has been used because it is the most popular network supporting SC and its documentation is easily accessible. Today, many new networks are being developed on the BC network. It is possible to use a BC network that supports another SC. In the future, with the new technological developments on BC, the new BC networks with more cost-effective may be developed. The DCMApp can be rebuilt on these networks. Thus, more cost-effective DCMApp can be developed.

The fact that users do not need any membership system when logging into the application indicates that the information such as user name and password is not stored at a central point. Users can open the application directly from their computers while using the application, and by entering the private key information of the crypto wallets into the application, they can perform private transactions on the public BC network.

Since the application is distributed and stores its data in the local database, it will not be affected by any interruptions. Because the agreements are stored in the Ethereum network, access to this information is possible from any point connected to the Ethereum network. The fact that detailed information is being transferred from a central repository to the application's local database can be seen as a weak point. However, an interruption in repositories will only prevent the retrieval of current information. With the information in the local database, the application will continue to run uninterruptedly. As an example, if the resource repositories, which we consider centrally, cannot serve, users will be able to continue to deal with the existing information from the local database in their applications. They will not be able to obtain information about the new status of the resources only. This can cause two different problems. In the first case, other users will not be aware of a newly added resource in the network. The system will continue to operate with old resource information in DCMApps' local databases. In the second case, the information of an existing resource can be changed or the resource can be passive. In this case, job offers will continue to be sent to this resource based on old information. The resource owner cannot share the current status of the resource with other users, so he does not have to accept job offers from the previous status. Thus, changing the source information will not prevent the operation of the system. Repository service can also be replicated with new servers if necessary. Against possible barriers, such as access restriction by governments, BC networks are capable of continuing to work uninterruptedly. It will be sufficient to have access to any point that can access the Ethereum network. It is therefore not possible for governments to restrict DCMApp.

DCMApp owes its ability to work in a distributed structure to the Ethereum network, as we said before. The Ethereum network is a very reliable network and it is almost impossible to change the operations performed on this network. In order to change past operations on the network, the number of nodes in the network must be at least 50% of the total number

**TABLE 1.** Cloud manufacturing vs BC supported cloud manufacturing.

| | Cloud Manufacturing | BC Supported Cloud Manufacturing |
|---|---|---|
| **Reliability** | As reliable as 3rd party | Provides the entire BC network |
| **Availability** | Depends on central server (s) | Depends on BC network |
| **Price** | The cost of servers running 24/7 | Charge only for transaction |
| **Development** | Easy | Harder |
| **Data Security** | Data can be modified | Data can never be changed |

of nodes. This attack is called 51% attack. If this attack is attempted, it is necessary to have more than half of the instant processing power for a long time. The processing power of the Ethereum network for the date of 24.06.2019 is 155TH/h. The average cost of obtaining 51% of this power for only 1 hour is $165,246. These values are obtained from the average values of known Ethereum mining companies. Thus, it is easily understood that storing data on the Ethereum network is much safer than storing it on a third party company's computer.

The cost of creating a JSC, it is calculated as approximately 0.02 ETH in the tests. The cost of submitting an offer is calculated as 0.01 ETH. Other processes were found to cost approximately 0.01 ETH. These values can be seen as high values for a small business, but there is a cost for a digital agreement to be made in a secure and distributed network. These agreements is not only stored on one or more computers, but also stored and operated on thousands of computers serving the Ethereum network. The transaction fee has been paid for storing these information on the Ethereum network forever. The fee for such a service that can store lifelong data is very low.

If we compare this work with a central cloud manufacturing platform, it is clear that the central application has failed in terms of security. A central application must run on a server belonging to a person or company. But, this has a cost. A finance must be incurred to develop, maintain and manage not only server costs but also the application. Since there is a commercial concern of a centralized system, a fee is charged to the traders. The pricing policy may vary by company. Some companies charge periodic fees, while others receive commissions per transaction. In the architecture proposed here, the fees paid are paid as the rental fees of the computers operating on the network.

The prominent items in the comparison of cloud manufacturing and BC supported cloud manufacturing are summarized in Table 1. In addition, the pros and cons of DCMApp are given in the Table 2.

- *Reliability:* In CM, users rely on intermediaries who present the application to users. The system is as reliable as much as the reliability of this broker. However, the BC-supported CM is not connected to a point, the entire BC network ensures the reliability of the system.
- *Availability:* CM runs on central servers. The uptime rate of the CM application depends entirely on the central server systems. In the BC supported CM applications, when the applications are run on the BC network, they will no longer depend on the central servers and will run on the BC network. In the event that any node in the BC network fails, the other nodes in the network will keep the network running.
- *Price:* CM applications must be running 24/7. A server is needed for applications to run continuously. The fee paid to the servers continues continuously for certain periods. In the BC supported CM, if the application is designed to run distributed on BC, only a fee will be paid for the transaction. There is no continuously paid fee for the survival of the BC network.
- *Development:* Since BC supported CM applications will become a dApp, the development of dApp is much more different and difficult than traditional application development processes.
- *Data Security:* Any person in the CM who has access to the database can change the data. The security of the data depends on the person who accesses the database. However, in BC supported CM is very difficult to change because the data is written to the blocks in the form of chains. This is almost impossible in public BC networks.

**TABLE 2.** Pros & Cons.

| **Pros** |
|---|
| Agreements between the parties are carried out without any intermediaries. |
| The manufacturing agreement determined by SC is inviolable. Any party may not go beyond the conditions specified in the contract. |
| Operations are irreversible. All operations are secured. |
| Since the existing public BC infrastructure is used, no infrastructure investment is required. |
| Efficient cash flow. Payments can be made between the parties through the SC using the crypto currency. |
| It cannot be restricted by governments. |
| Zero downtime. Distributed cloud manufacturing application is always on. |
| **Cons** |
| In order to expand the application, it is necessary to know the solidity language besides the software language used in the application. |
| Transparent. Since the data of the agreements are held in a public network, it is possible to read them. However, they are only those who can extract data from the BC network. This requires expertise. |
| Operations are irreversible. It is impossible to correct accidental operations. |
| The transaction fee must be paid for each write operation. |

The developed application can operate in a distributed manner on a continuous living network. A single version of the software prepared in this architecture does not have to be used. Different versions can be used by users in the Ethereum network at the same time. With the new version of the code, users who want to benefit from the improvements will be able to update their applications for new agreements. Since the application is offered as open source, users can continue to use the system with different versions by changing the codes as they wish. It would be very exciting to have an application that can be continuously developed and accessible at any time, regardless of the person or organization. In central applications, it is mandatory to use the version offered by the company. The history of the agreements to be made will be up to the life of the company.

## VI. CONCLUSION

With the capabilities of cloud manufacturing, resource owners and resource-seeking users can easily share resources through platforms. However, the reliability of the platforms is seen as a question mark for the users. In this study, a BC based application is introduced for users to make agreements among themselves without the need of any third party. The originality of this study is that the agreements are realized through the Ethereum network, which is a public BC network and supports SC. However, the application is designed in a hybrid structure. Thanks to the hybrid structure, users pay only for agreements that need to be secured. If the application was run in a completely public structure, all information would have to be stored on the public network, which would incur a serious cost. Apart from all this, in private networks, the fact that the network is controlled by a person or organization raises serious questions for users. In addition, the server infrastructure must be provided by a party for private networks. For these reasons, a hybrid structure was preferred. With this model, users will be able to make manufacturing agreements between each other and make their payments without any intermediaries. In this study, the use of BC technology in cloud manufacturing is attempted to be explained with an example application and it is aimed to shed light on the future studies. For future work, designed SCs may be developed and other services, such as machining as a service, will be directly integrated into the system.

## REFERENCES

[1] D. Wu, D. Rosen, and D. Schaefer, *Cloud-Based Design Manufacturing: Status Promise*. Cham, Switzerland: Springer, Jul. 2014, pp. 1–24.

[2] K. Zhou, T. Liu, and L. Zhou, "Industry 4.0: Towards future industrial opportunities and challenges," in *Proc. 12th Int. Conf. Fuzzy Syst. Knowl. Discovery (FSKD)*, Aug. 2015, pp. 2147–2152.

[3] Y. Liu and X. Xu, "Industry 4.0 and cloud manufacturing: A comparative analysis," *J. Manuf.Sci. Eng.*, vol. 139, no. 3, Oct. 2016, Art. no. 034701, doi: 10.1115/1.4034667.

[4] Y. Lu, "Industry 4.0: A survey on technologies, applications and open research issues," *J. Ind. Inf. Integr.*, vol. 6, pp. 1–10, Jun. 2017.

[5] H. Lasi, P. Fettke, H.-G. Kemper, T. Feld, and M. Hoffmann, "Industry 4.0," *Business*, vol. 6, no. 4, pp. 239–242, Aug. 2014.

[6] J. Lee, B. Bagheri, and H.-A. Kao, "A cyber–physical systems architecture for industry 4.0-based manufacturing systems," *Manuf. Lett.*, vol. 3, pp. 18–23, Jan. 2015. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S221384631400025X

[7] D. Georgakopoulos, P. P. Jayaraman, M. Fazia, M. Villari, and R. Ranjan, "Internet of Things and edge cloud computing roadmap for manufacturing," *IEEE Cloud Comput.*, vol. 3, no. 4, pp. 66–73, Jul. 2016.

[8] J. Wan, S. Tang, Z. Shu, D. Li, S. Wang, M. Imran, and A. V. Vasilakos, "Software-defined industrial Internet of Things in the context of industry 4.0," *IEEE Sensors J.*, vol. 16, no. 20, pp. 7373–7380, Oct. 2016.

[9] W. He and L. Xu, "A state-of-the-art survey of cloud manufacturing," *Int. J. Comput. Integr. Manuf.*, vol. 28, no. 3, pp. 239–250, Mar. 2015.

[10] G. Skulj, R. Vrabic, P. Butala, and A. Sluga, "Decentralised network architecture for cloud manufacturing," *Int. J. Comput. Integr. Manuf.*, vol. 30, nos. 4–5, pp. 395–408, 2017, doi: 10.1080/0951192X.2015.1066861.

[11] B. Li, L. Zhang, S.-L. Wang, F. Tao, J. Cao, X.-D. Jiang, X. Song, and X.-D. Chai, "Cloud manufacturing: A new service-oriented networked manufacturing model," *Comput. Integr. Manuf. Syst.*, vol. 16, no. 1, pp. 1–7, Jan. 2010.

[12] G. Adamson, L. Wang, M. Holm, and P. Moore, "Cloud manufacturing—A critical review of recent development and future trends," *Int. J. Comput. Integr. Manuf.*, vol. 30, nos. 4–5, pp. 347–380, 2017, doi: 10.1080/0951192X.2015.1031704.

[13] L. Bo-Hu, Z. Lin, W. Shi-Long, T. Fei, C. Jun-Wei, J. Xiao-Dan, S. Xiao, and C. Xu-Dong, "Cloud manufacturing: A new service-oriented networked manufacturing model," *Comput. Integr. Manuf. Syst.*, vol. 16, no. 1, pp. 1–7, 2010.

[14] L. Zhang, Y. Luo, F. Tao, B. H. Li, L. Ren, X. Zhang, H. Guo, Y. Cheng, A. Hu, and Y. Liu, "Cloud manufacturing: A new manufacturing paradigm," *Enterprise Inf. Syst.*, vol. 8, no. 2, pp. 167–187, 2014, doi: 10.1080/17517575.2012.683812.

[15] L. Wang, X. V. Wang, L. Gao, and J. Váncza, "A cloud-based approach for WEEE remanufacturing," *CIRP Ann.*, vol. 63, no. 1, pp. 409–412, 2014. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0007850614001176

[16] L. Ren, L. Zhang, L. Wang, F. Tao, and X. Chai, "Cloud manufacturing: Key characteristics and applications," *Int. J. Comput. Integr. Manuf.*, vol. 30, no. 6, pp. 501–515, Jun. 2017, doi: 10.1080/0951192x.2014.902105.

[17] M. H. Mourad, A. Nassehi, D. Schaefer, and S. T. Newman, "Assessment of interoperability in cloud manufacturing," *Robot. Comput.-Integr. Manuf.*, vol. 61, Feb. 2020, Art. no. 101832.

[18] A. N. Samant, V. S. Narwane, and I. A. Siddavatam, "The critical assessment of cloud manufacturing technology and future trends," in *Proc. Int. Conf. Emanations Mod. Technol. Eng.*, New Delhi, India, Mar. 2017, vol. 5, no. 3, pp. 84–87. [Online]. Available: https://ijritcc.org/download/conferences/ICEMTE_2017/Track_3_(MECH)/1487918432_24-02-2017.pdf

[19] P. Wang, R. Gao, and Z. Fan, "Cloud computing for cloud manufacturing: Benefits and limitations," *J. Manuf. Sci. Eng.*, vol. 137, no. 4, Aug. 2015, Art. no. 040901.

[20] L. Ren, L. Zhang, F. Tao, C. Zhao, X. Chai, and X. Zhao, "Cloud manufacturing: From concept to practice," *Enterprise Inf. Syst.*, vol. 9, no. 2, pp. 186–209, Feb. 2015, doi: 10.1080/17517575.2013.839055.

[21] D. Contractor and D. Patel, "Accountability in cloud computing by means of chain of trust," *Int. J. Netw. Secur.*, vol. 19, pp. 251–259, Jan. 2017.

[22] M. Hamdaqa and L. Tahvildari, *Cloud Computing Uncovered: A Research Landscape* (Advances in Computers), vol. 86, A. Hurson and A. Memon, Eds. Amsterdam, The Netherlands: Elsevier, 2012, pp. 41–85. [Online]. Available: http://www.sciencedirect.com/science/article/pii/B9780123965356000028

[23] A. V. Barenji, H. Guo, Z. Tian, Z. Li, W. M. Wang, and G. Q. Huang, "Blockchain-based cloud manufacturing: Decentralization," in *Proc. 25th Int. Conf. Transdisciplinary Eng. (ISTE)*. Rome, Italy: Transdisciplinary Engineering Methods for Social Innovation of Industry 4.0, Jun. 2018, pp. 1003–1011. [Online]. Available: http://ebooks.iospress.nl/publication/49887

[24] M. Jammal, H. Hawilo, A. Kanso, and A. Shami, "Mitigating the risk of cloud services downtime using live migration and high availability–aware placement," in *Proc. IEEE Int. Conf. Cloud Comput. Technol. Sci. (CloudCom)*, Dec. 2016, pp. 578–583.

[25] S. Abeyratne and R. Monfared, "Blockchain ready manufacturing supply chain using distributed ledger," *Int. J. Res. Eng. Technol.*, vol. 5, no. 9, pp. 1–10, 2016.

[26] Z. Li, A. V. Barenji, and G. Q. Huang, "Toward a blockchain cloud manufacturing system as a peer to peer distributed network platform," *Robot. Comput.-Integr. Manuf.*, vol. 54, pp. 133–144, Dec. 2018. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S073658451830022X

[27] B. M. Lavanya, "Blockchain technology beyond bitcoin: An overview," *Int. J. Comput. Sci. Mobile Appl.*, vol. 6, no. 1, pp. 76–80, 2018. [Online]. Available: http://www.ijcsma.com

[28] S. Nakamoto. (2009). *Bitcoin: A Peer-to-Peer Electronic Cash System*. [Online]. Available: http://www.bitcoin.org/bitcoin.pdf

[29] M. Crosby, P. Pattanayak, S. Verma, and V. Kalyanaraman, "Blockchain technology: Beyond bitcoin," *Appl. Innov.*, vol. 2, nos. 6–10, p. 71, 2016. [Online]. Available: http://scet.berkeley.edu/wp-content/uploads/AIR-2016-Blockchain.pdf

[30] (2019). *Ethereum Market Capital*. Accessed: Jun. 26, 2019. [Online]. Available: https://coinmarketcap.com/currencies/ethereum/

[31] G. Karame, E. Androulaki, and S. Capkun, "Double-spending fast payments in bitcoin," in *Proc. ACM Conf. Comput. Commun. Secur.*, Oct. 2012, pp. 906–917.

[32] G. O. Karame, E. Androulaki, and S. Capkun, "Two bitcoins at the price of one? Double-spending attacks on fast payments in bitcoin," *IACR Cryptol. ePrint Arch.*, vol. 2012, p. 248, Oct. 2012.
[33] W. Cai, Z. Wang, J. B. Ernst, Z. Hong, C. Feng, and V. C. M. Leung, "Decentralized applications: The blockchain–empowered software system," *IEEE Access*, vol. 6, pp. 53019–53033, 2018.
[34] A. Bahga and V. Madisetti, "Blockchain platform for industrial Internet of Things," *J. Softw. Eng. Appl.*, vol. 9, pp. 533–546, Jan. 2016.
[35] CryptoKitties. *CryptoKitties*. [Online]. Available: https://www.cryptokitties.co/
[36] M. Sevilla, "Does blockchain hold the key to a new age in supply chain transparency and trust?" Capgemini Res. Inst., Paris, France, Tech. Rep., 2018. [Online]. Available: https://www.capgemini.com/research/does-blockchain-hold-the-key-to-a-new-age-in-supply-chain-transparency-and-trust/
[37] Z. Zheng, S. Xie, H. Dai, X. Chen, and H. Wang, "An overview of blockchain technology: Architecture, consensus, and future trends," in *Proc. IEEE Int. Congr. Big Data (BigData Congress)*, Jun. 2017, pp. 557–564.
[38] M. Armbrust, A. Fox, R. Griffith, A. D. Joseph, R. Katz, A. Konwinski, G. Lee, D. A. Patterson, A. Rabkin, I. Stoica, and M. Zaharia, "A view of cloud computing," *Commun. ACM*, vol. 53, pp. 50–58, Apr. 2010.

**BARAN KAYNAK** received the B.S. and M.S. degrees in industrial engineering from Sakarya University, Sakarya, Turkey, in 2011 and 2013, respectively, where he is currently pursuing the Ph.D. degree in industrial engineering.

He joined the Computer Research and Application Center, Sakarya University, in 2011. His current research interests include software system development, cloud manufacturing, and blockchain.

**SÜMEYYE KAYNAK** received the B.S., M.S., and Ph.D. degrees in computer engineering from Sakarya University, Sakarya, Turkey, in 2012, 2014, and 2019, respectively.

Since 2012, she has been a Research Assistant with the Department of Computer Engineering, Sakarya University. Her research interests include energy, software system development, blockchain, and artificial intelligence.

**ÖZER UYGUN** received the B.Sc. degree in industrial engineering, in 1999, and the M.Sc. and Ph.D. degrees in industrial engineering from Sakarya University, Turkey, in 2002 and 2008, respectively. He started his academic position at Marmara University and worked as a Lecturer, from 2000 to 2003. He was a Research Assistant with Sakarya University, from 2003 to 2008, where he is currently an Associate Professor Doctor. He was a Researcher in EU FP6 Network of Excellence (I*PROMS: 2004–2009) and FP6 STREP Project (IWARD: 2007–2009). He entered the Industrial Engineering Department, Sakarya University, in 1994. He has successfully completed the EFQM Assessor Training in Brussels, in 2015.