BLOCKCHAIN-ENABLED INFORMATION AS A SERVICE (IAAS) AND OPTIMAL FULFILLMENT CAPACITY BALANCING IN CYBER PLATFORM-DRIVEN CROWDSOURCED MANUFACTURING

A Dissertation
Presented to
The Academic Faculty

by

Mulang Song

In Partial Fulfillment
of the Requirements for the Degree
Master in the
George W. Woodruff School of Mechanical Engineering

Georgia Institute of Technology
May 2022

BLOCKCHAIN-ENABLED INFORMATION AS A SERVICE (IAAS) AND OPTIMAL FULFILLMENT CAPACITY BALANCING IN CYBER PLATFORM-DRIVEN CROWDSOURCED MANUFACTURING

Approved by:

Dr. Roger J. Jiao, Advisor
School of Mechanical Engineering
Georgia Institute of Technology

Dr. Shuman Xia,
School of Mechanical Engineering
Georgia Institute of Technology

Dr. J. Rhett Mayor,
School of Mechanical Engineering
Georgia Institute of Technology

Dr. Roxanne Moore,
School of Mechanical Engineering
Georgia Institute of Technology

Date Approved: April 6, 2022

ACKNOWLEDGEMENTS

The work of this thesis would be impossible to finish without the help and support of lots of people and the great research environment provided by the Georgia Institute of Technology.

First, thanks to my advisor Dr. Roger Jiao at Georgia Institute of Technology, for providing guidance, constructive advice, and encouragement during my graduate studies. Secondly, I want to express my sincere gratitude to the thesis reading committee members, Dr. Roger Jiao, Dr. J. Rhett Mayor, Dr. Roxanne Moore, and Dr. Shuman Xia, for their help and time in revising the thesis work. Thirdly, I would like to thank all the students studying in GTMI 264B, Dr. Xuejian Gong, Pan Zou, Shu Wang, Jianyuan Peng, Yiyun (Cindy) Fei, and Roosan Liyoons, and the staff at Georgia Institute of Technology. Lastly, I would like to thank my family for their great support during my graduate studies.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF SYMBOLS AND ABBREVIATIONS

IaaS	Information as a Service
MaaS	Manufacturing as a Service
ICT	Information and Communications Technologies
IoTs	Internet of Things
CNs	Customer Needs
DPs	Design Parameters
PVs	Process Variables
PoW	Proof of Work
PoS	Proof of Stake
PoA	Proof of Authority
IPFS	Interplanetary File System
CID	Content Identifier
ECC	Evolutionary Competition-Cooperation
C^0	Customer order
M	Manufacturing agents
α, A	Manufacturing agent cluster index and total number
$\mu_{n_\alpha}^{\alpha}$	Bidding manufacturing agent n_α in cluster α
φ_n	Non-bidding agents
D^0	Design specs
P^0	Process specs
δ_k, Δ	Manufacturing subtask and its associated product
c_i	i_{th} cluster of manufacturers

η_i	Fulfillment capacity factor of i_{th} cluster
$F_i(t)$	Participation level of manufacturing cluster c_i
ub_i	Uncorrected bidding cost of manufacturing cluster c_i
b_i	Bidding cost of manufacturing cluster c_i
ρ_i	Crowdsourcing income of manufacturing cluster c_i
P	Uncorrected crowdsourcing income
E_i	Manufacturing income of manufacturing cluster c_i
C	Strategy C, participate in the bidding.
D	Strategy D, don't participate in the bidding
f_i^C	Average payoff of choosing strategy C of manufacturing cluster c_i
f_i^D	Average payoff of choosing strategy D of manufacturing cluster c_i
r_i	Replicator equation of manufacturing cluster c_i
N_i	Number of manufacturers in cluster i
j_i	Number of participating manufacturers in cluster i
$T_{j_i}^+$	Probability that the number of participating manufacturers in cluster i changes from j_i to $(j_i + 1)$
$T_{j_i}^-$	Probability that the number of participating manufacturers in cluster i changes from j_i to $(j_i + 1)$ from j_i to $(j_i - 1)$
f_{C_i}	Fitness function of selecting strategy C in ith manufacturing cluster
f_{D_i}	Fitness function of selecting strategy D in ith manufacturing cluster
w	Selection Intensity
$t_{a\to N_i}$	Probability of number of participating manufacturers in i_{th} cluster j_i changes from a to N_i
$p_f^i, t_{0\to N_i}$	Probability of number of participating manufacturers in i_{th} cluster j_i changes from 0 to N_i
r_i^{new}	Replicator equation of cluster c_i with the substitution of new variables

$b_{i,min}, b_{i,max}$	Minimum and Maximum of bidding cost b_i
g_i	Fraction of total manufacturers for cluster c_i

SUMMARY

As a new emerging manufacturing paradigm, platform-driven crowdsourced manufacturing utilizes the cooperation between the platform, designer, and service providers to configure and fulfill the supply chain. In this value creation and delivery process, the cyber platform enables and manages the interaction between each participant in the supply chain to respond to varieties of customer needs which lets platform-driven crowdsourced manufacturing become a persuasive approach to seeking manufacturing solutions.

This thesis examines platform-driven crowdsourced manufacturing based on two unique perspectives: Information as a Service (IaaS) fulfillment and operational excellence of the platform. From the first perspective, this thesis analyzes the use case of the cyber platform in the platform-driven crowdsourced manufacturing system based on its workflow. An IaaS fulfillment system is designed based on the analysis using blockchain and distributed file-sharing technologies. The proposed system is distributed, which fulfills IaaS by providing secured information upload, sharing, and management services. The decentralization feature of the system reduces the cost of trust for using the system. From the perspective of operational excellence, the thesis models the interactions between users and their decision-making process in the system based on ECC game theory, population dynamics, and the Moran process. Based on the models, an optimization strategy is proposed to manage the fulfillment capacity balance by facilitating the participation level of users.

CHAPTER 1. INTRODUCTION

The transition in the market causes new challenges to manufacturing companies where they are forced to increase and manage more product varieties and complexities to satisfy the dynamics of customer needs (Brettel et al., 2014). Confronting new challenges, the ability to design and test new products effectively becomes the unique key to success in the competition (Jiao et al., 2003). Under the transformation to the buyer's market, larger product variety and complexity affect both design and manufacturing domains, attracting attention in the design domain from academia and industry.

Transformation in the market leads the transformation in the industry. To confront the challenges and intense competition brought by demands in highly customized products with reduced life cycles, manufacturing companies integrate the newly emerging technologies, new design strategies, and new organizational structures.

1.1 Platform-driven Crowdsourced Manufacturing

As an emerging open business model, crowdsourcing allows the business owner to utilize the extra resource and capabilities of the others across the crowd to finish a certain task. Four key elements in a crowdsourcing system are identified as the crowd, the crowdsourcer, the crowdsourcing task, and the crowdsourcing platform (Hosseini et al., 2014). The crowd is an essential component of the system for participating in crowdsourcing tasks. The crowdsourcer is the entity in a crowdsourcing system that outsource task across groups of crowds for seeking solutions (Bücheler and Sieg, 2011). The crowdsourcing task is an activity in which the crowdsourcer participates. The

crowdsourcing platform gathers active users in a virtual platform for involving value-creating activities which collaborate resources and capabilities from distributed individuals (Kohler, 2015).

From a special perspective, platform-driven crowdsourced manufacturing is a crowdsourcing system that enables product innovation and production after the coordination and negotiation across the crowdsourcer in a cyber platform (Gong et al., 2021). As an open manufacturing model, platform-driven crowdsourced manufacturing enables the utilization of external knowledge and resources for achieving product fulfillment collaboratively for a manufacturer in a distributed crowdsourcing network. In this regard, key elements in platform-driven crowdsourced manufacturing are reidentified as the crowdsourcer, crowdsourced manufacturing tasks, and cyber platform. The crowdsourcer includes both individuals who initiate a new manufacturing task and individuals who provide services in the product fulfillment process. The crowdsourced manufacturing task is a task initiated by the crowdsourcer which can be decomposed into several processes and fulfilled by the service-providing crowdsourcer. The cyber platform provides the virtual space for accruing crowds and provides necessary services to crowdsourcers in the supply chain formulation and product fulfillment process. For a crowdsourcing framework, four prerequisites are identified cognitive diversity, independence, decentralization, and aggregation (Surowiecki, 2004). For platform-driven crowdsourced manufacturing, the cyber platform should take the responsibility of management to meet the prerequisite of facilitating prosperity for crowdsourced manufacturing through providing services.

In cyber platform-driven crowdsourced manufacturing, there are three fundamental issues which are manufacturing activities digitalization, information coordination and management, and operational management. The term blockchain-enabled information as a service in the title is derived from these two issues. Optimal fulfillment capacity balancing corresponds to the third issue.

1.2 Information as a Service for Platform-driven Crowdsourced Manufacturing

With the vision of open manufacturing, X as a service (XaaS) is the term that represents the service provided in X of the function domain (Kusiak, 2020). In platform-driven crowdsourced manufacturing, the value is created and fulfilled through the interaction between the crowdsourcer and the cyber platform, which are two agents in the system. The cyber platform attracts and gathers the crowdsourcer and provides necessary services to the user. In this regard, the cyber platform in platform-driven crowdsourcing manufacturing has become a service-oriented paradigm. The cyber platform is able to facilitate the fulfillment of manufacturing-as-a-service (MaaS) in crowdsourced manufacturing by providing decision support services as intelligent cognitive assistants (ICA) to manufacturers (Gong et al., 2021). Complementarily, the fulfillment of information-as-a-service is fulfilled through the collaborative product fulfillment process in crowdsourced manufacturing by serving the service of information exchanging and sharing among crowdsourcers.

Information exchanging and sharing are significant in the product fulfillment process in an open manufacturing network. Information sharing and coordination in the supply chain can reduce costs and increase the overall economic benefit (Sahin & Robinson, 2005).

It brings direct effect to the stakeholder in the supply chain. In platform-driven crowdsourced manufacturing, information exchanging and sharing is necessary for the decision strategy coordination and product fulfillment. Furthermore, it establishes the fundamental requirement for IaaS fulfillment in the cyber platform.

Different from the traditional manufacturing paradigm, in crowdsourced manufacturing, manufacturers and logistic service providers are widely dispersed and connected through the cyber platform or the information system. One issue that existed in the IaaS fulfilled by the system or platform is data transparency and privacy. In the product fulfillment process, the management of data transparency and privacy can have the trust issue due to distributed manufacturing layout. For the centralized system, the main trust provider needs to be identified. For example, a centralized information system is managed by a third-party agent. Blockchain technologies and other distributed file-sharing technologies can be considered the solution to this problem. The cooperation between agents in a crowdsourced task can be solidified by this kind of security database. The stored data is distributed, traceable, and unerasable without the management of a third party. The decentralization feature of the blockchain technologies make it more suitable for providing data management services to a distributed manufacturing system.

1.3 Fulfillment Capacity Balancing in Crowdsourced Manufacturing

In the cyber platform-driven crowdsourced manufacturing, the crowdsourced task is fulfilled by manufacturers that provide different kinds of services. Multiple clusters of manufacturers and other service providers are formed in the platform. The match and

balance of fulfillment capacity become one of the fundamental issues for platform prosperity.

The solution-seeking process in crowdsourced manufacturing involves both cooperation and competition. Despite the cooperation that happened in the fulfillment process, in the supply chain formulation process of crowdsourced manufacturing, manufacturers are supposed to participate in steps of the product fulfillment process based on their manufacturing capabilities and capacity. For manufacturers that have similar manufacturing capabilities, competition relationships are unavoidable in such a manufacturing cluster as the result of limited crowdsourced tasks. Since a successful crowdsourcing platform requires not only the number of crowdsourced tasks but also requires high participation level from crowdsourcers (Thuan et al., 2015), the term fulfillment capacity is introduced here as the total manufacturing capacity of a manufacturing cluster in platform-driven crowdsourced manufacturing which reflect a certain type of production capacity in the platform.

Increasing and balancing different kinds of fulfillment capacity is one of the key factors for fulfilling crowdsourced tasks. From the perspective of the platform, this can be achieved through a management protocols strategy. This requires an understanding of the behavior and changes in the crowdsourcing system and the prediction methodology of the changes in the system.

1.4 Technical Challenges and Research Tasks

The objective of the work is to investigate platform-driven crowdsourced manufacturing from two unique perspectives. The first perspective is how to design

information as a service (IaaS) fulfillment system for the product fulfillment process in the crowdsourced task. The second perspective is how to develop an optimal management strategy based on the population dynamics model to manage and optimize fulfillment capacity between clusters of manufacturers, which also inspires the participation willingness in the tournament-based bidding process of the contracting process in platform-driven crowdsourcing manufacturing.

For the first objective, the technical challenges come from three research questions: (i) How to determine the information flow in the cyber platform; (ii) How to exchange and share the information across the different agents of crowds; (iii) How to manage the information with lower cost the higher security. For the second objective, the technical challenges are (i) how to model the interactive behavior among the manufacturers in a tournament-based bidding process for a crowdsourced task; (ii) how to simulate the stochastic process in the simulation of the evolution of manufacturing clusters; (iii) how to design the optimal strategy based on models.

1.5 Organization of This Thesis

The rest of this thesis work is organized as shown in figure 1-1. Chapter 2 reviews the related work of smart manufacturing, cyber physical system, cyber platform, crowdsourcing product fulfillment and blockchain-related technology. Chapter 3 presents the analysis and design of a blockchain-enabled IaaS fulfillment system for product fulfillment crowdsourcing which includes the use case analysis and architecture design. Chapter 4 discusses IaaS management provided by the proposed system from the perspective of user management, smart contract management, and information

management. Chapter 5 shows an application of the proposed system based on a tank trailer case. The illustration is focused on the information flow management in the product fulfillment process of the tank trailer. Chapter 6 discusses fulfillment capacity balancing and optimization in cyber platform-driven crowdsourced manufacturing. It introduces a population dynamics model and Moran process based on evolutionary competition-cooperation (ECC) game theory. An optimization strategy is also proposed and illustrated by a case study example in this chapter. Chapter 7 summarizes the contributions, assumptions, and limitations of this thesis work and provides a future vision of improvements.

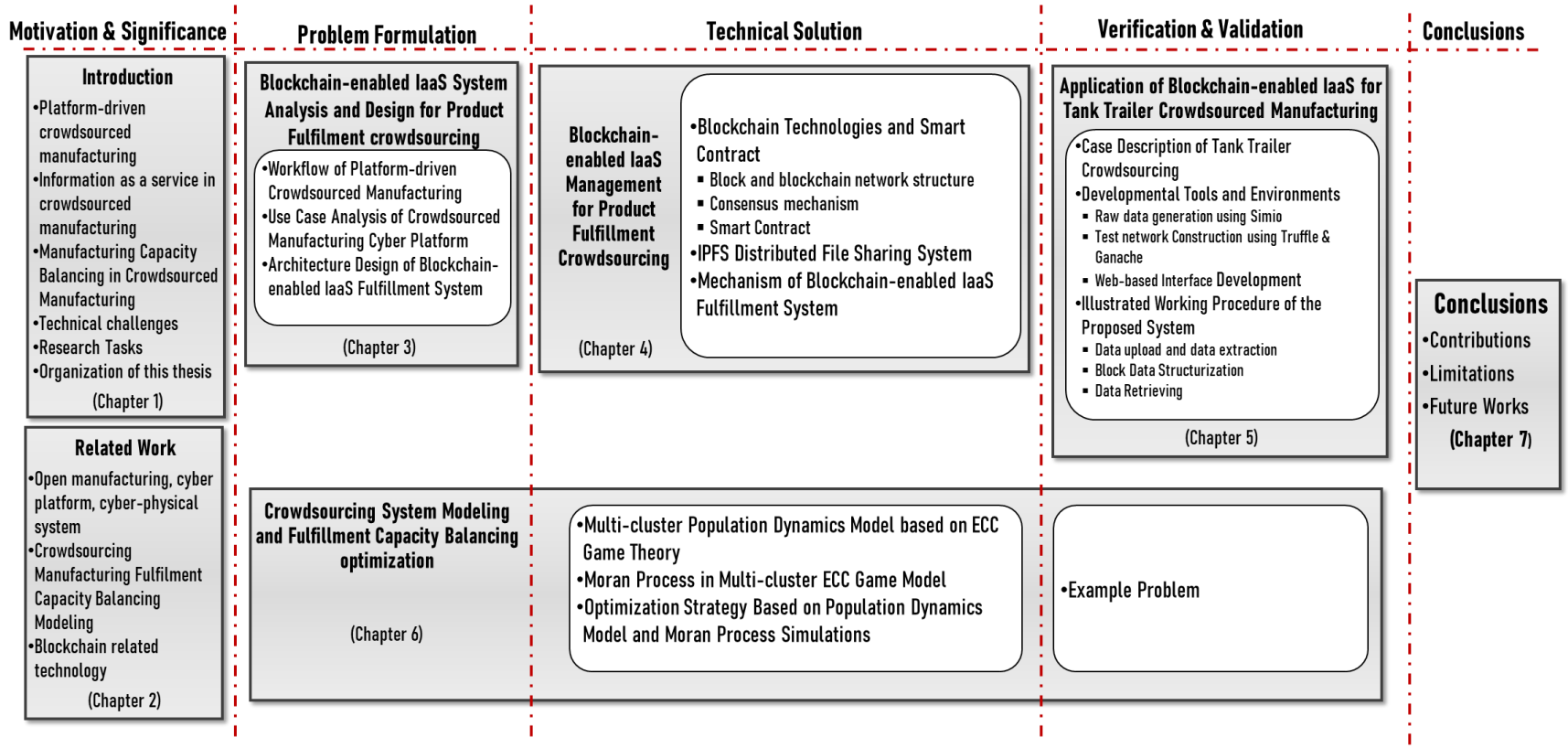


Figure 1-1 Technical Roadmap of the Thesis

CHAPTER 2. RELATED WORK

2.1 Smart Manufacturing, IoT, Cyber Platform, and the Cyber-Physical System

New challenges and new technology bring revolution to the industry. The manufacturing ecosystem can be divided into Industrial 1.0, 2.0, 3.0, and 4.0, which go through the process from centralized organization to network or even decentralized organization (Li et al., 2018). Breakthroughs of technology bring innovations in the manufacturing system, which leads to the emergence of the automated, computerized and complex smart manufacturing system. The term smart manufacturing has attracted attention from both academia and industries. The smart manufacturing system is more open, which tends to have properties like stronger external connectivity and more manufacturing and resource sharing (Kusiak, 2017). These technology integrations in smart manufacturing inspire the design of the system that fulfills IaaS in cyber platform-driven crowdsourced manufacturing.

Open manufacturing is a paradigm of the smart manufacturing system that is designed for sharing knowledge, resources, and service in the manufacturing ecosystem, which presents a framework that provides service of information-sharing services (Li et al., 2018). Followed by the concept of smart manufacturing, distributed manufacturing is a paradigm that utilizes information and communication technologies (ICT) to achieve the manufacturing value chain through a decentralized manufacturing network (Srai et al., 2016). Social manufacturing is another emerging paradigm of smart manufacturing which combine the usage of the Cyber-Physics System (CPS) with social media to provide services in design and production (Jiang et al., 2017).

Cloud manufacturing utilizes a service-oriented platform to manage the manufacturing capabilities and resources across the internet to provide manufacturing services to users, which has become a new paradigm of smart manufacturing systems (Zhang et al., 2012). Qu et al. proposed an IoT-based real-time production logistic synchronization system for smart cloud manufacturing in 2015, which integrates IoT to cloud manufacturing to synchronize the dynamics in production. In the proposed system, information in real-time collected from sensors and RFID tags flows through a multi-level system for logistic production synchronization. The information flow in the virtual space corresponds to the material flow in the real world in the proposed system.

It can be summarized that ICT, IoT, and CPS are the most common technology adopted in the new manufacturing system, which ensures the establishment of a service-oriented cyber platform for information, logistic, resources coordination, and management. These technologies enable a new level of architecture design of information systems for fulfilling IaaS in the production of the cyber platform-driven crowdsourced manufacturing system.

2.2 Crowdsourcing Manufacturing Fulfillment Capacity Balancing Modeling

As mentioned earlier, fulfillment capacity refers to the total manufacturing capacity of a manufacturing cluster in a crowdsourcing manufacturing system. The fulfillment capacity could be evaluated from two perspectives which are level and balance across different types of product capacities. To come up with a strategy of management, modeling the interaction and behaviors of agents in crowdsourcing manufacturing is necessary.

Guazzini et al. provided a mathematical model of crowdsourcing in 2015. The model includes a preference for collaboration simplicity of the tasks to describe the behavior in groups of crowdsourcers. The performance and effectiveness of crowdsourcing are evaluated through the fitness of crowdsourcers. The model also indicates that the effectiveness and performance of crowdsourcing can be optimized based on the number of groups. Hoßfeld et al. proposed another crowdsourcing modeling method to predict growth dynamics through measurement-based statistical analysis in 2011. The population dynamics model is also utilized to describe the dynamics in the platform.

In the cyber platform-driven crowdsourcing manufacturing, modeling the competition relationship and cooperation relationship inside and among manufacturing clusters becomes more significant. Population dynamics becomes a powerful tool that is able to demonstrate mechanisms in evolutionary group interactions (Perc et al., 2013). In 2015, Chen et al. studied competition and cooperation modeling in public goods games with different punishment strategies.

There are lots of studies in evolutionary cooperation and competition game theory. A colloquium discusses the evolutionary games on with the condition of different kinds of multilayer networks (Wang et al., 2015). Studies in multiplayer ECC game theory imply that the difficulties and complexities increase extremely when it involves multiple players and strategies (Gokhale & Traulsen, 2010). Additionally, in complex multiplayer games, simulation based on graph theory can demonstrate interactions among players, which can validate the results of the evolutionary dynamics of populations in an analytic model (Pena et al., 2016).

These studies provide the theoretical basis for modeling the evolutionary dynamics of fulfillment capacity based on the growth of manufacturing clusters and corresponding validation through simulations. Furthermore, parameterization of the dynamics model enables the management strategy based on optimization.

2.3 Blockchain Technologies Applications

The concept of Blockchain was firstly introduced firstly in 2008 by Satoshi Nakamoto as the fundamental technology of the digital currency bitcoin. Blockchain technology is commonly implemented in digital currency (Nderwood, 2016). Despite the financial area, blockchain has been introduced in different industries.

Software solutions could be offered by employing blockchain as a software connector, which can provide interaction services across different software blocks. It can be applied in communication services, coordination services, and facilitation services (Xu et al., 2016). Transparency and traceability make blockchain suitable for the manufacturing supply chain, which has huge potential for supply chain function transformation (Abeyratne & Monfrad, 2016 & Dutta et al., 2020). Tian et al. proposed a traceability system based on blockchain technology for tracking agri-food safety in 2016, which can provide services of information tracing, freshness checking. In 2021, Liu et al. proposed a smart tracking and tracing platform for the drug supply chain using blockchain technology.

Transparency and security in information sharing and exchanging can be fulfilled by the blockchain-enabled cyber platform by providing unified standards and protocols (Jiang et al., 2021). The blockchain-based platform is also designed for manufacturing. With the implementation of IoT, a peer-to-peer platform called BPIIoT is proposed for

manufacturing. The proposed platform allows the development of distributed apps (Dapps) that can enhance the existing cloud-based manufacturing (Bahga & Madisetti, 2016). The Dapps is also known as smart contacts, which is the self-executable script stored on the blockchain. The automation created by smart contracts can facilitate the sharing of services, automate the cryptographically veritable workflows (Christidis et al., 2016).

CHAPTER 3. BLOCKCHAIN-ENABLED IAAS SYSTEM ANALYSIS AND DESIGN FOR PRODUCT FULFILLMENT CROWDSOURCING

3.1 Workflow of Platform-driven Crowdsourced Manufacturing

To design an information service system to fulfill IaaS in cyber platform-driven crowdsourced manufacturing system, the workflow in such a manufacturing system is necessary to be clarified for establishing an understanding of behavior and functional requirements for crowdsourcers and the cyber platform. Figure 3-1 illustrates an adapted example of crowdsourced manufacturing workflow (Gong et al., 2022).

There are three physical domains in the figure which are the designer domain, crowdsourced manufacturing platform domain, and open manufacturing domain. The crowdsourced task is fulfilled through the activities achieved by the collaboration of participants in all three physical domains. In the open innovation domain, the designer D is the crowdsourcer that designs a product and proposes the crowdsourced task. Crowdsourced manufacturing platform domain store and restructure the design into crowdsourced tasks for bidding. There are three types of platform agents, where $\bar{P} = P^C \cup \bar{P}^I \cup \bar{P}^E$. It also provides evaluation services at the end of the product fulfillment process. The open manufacturing domain contains manufacturing agents that are clustered based on their manufacturing capabilities. For each manufacturing cluster, not all manufacturing agents participate in the bidding process. To start with, the designer design new products based on customer orders that saved into C^0 based on the customer needs CNs. Once the

design process is finished, designers can initialize the crowdsourcing process by sending the design specifications (DPs) of products to D^0 in the virtue field of crowdsourcing information management. In the database of design specifications, $\Delta = \delta_1 \times \dots \delta_q \times \dots \delta_Q$ denotes the product structure. In the equation, δ_q represents a manufacturing subtask of the product and $q \in [1, Q]$, where q indicates the index of subtask among Q number of subtasks in total. P^C represents the manufacturing configuration manager who receives the D^0 and restructures the design to manufacturing subtasks. Invitation broker $\bar{P}^I$ receives subtasks and broadcasts each subtask to the corresponding manufacturing clusters for bidding. Invitation broker $\bar{P}^I$ consists of individual invitation broker P_α^I, where $\forall P_\alpha^I \in \bar{P}^I$. The index α corresponds to manufacturing cluster α. Evaluation broker $\bar{P}^E$ follows a similar pattern which is composed of individual evaluation broker P_α^E. The evaluation broker evaluates the results of the bid and chooses the winner of each manufacturing cluster for participating in the crowdsourced task. In the bidding process, manufacturing agents μ_n^α in each manufacturing cluster $\overline{\mu^\alpha}$ proposes their manufacturing process as bids $B = \{B_1, \dots, B_\alpha, \dots, B_\mathrm{A}\}$ to the evaluation broker. After the evaluation, the crowdsourced manufacturing supply contract $S = \mu^{1^*} \times \dots \mu^{\alpha^*} \times \dots \mu^{A^*}$ is established, which contains each manufacturing agent. Manufacturing configuration manager P^C send process specification set $B^* = \{B_\alpha^*\}|_\mathrm{A}$ to process specifications database P^0 based on the returned manufacturing supply contract S. Final products are returned to the designer D through the product fulfillment process.

Material flow and knowledge flow are defined in the workflow, and the platform coordinates these flows back and forth between the open innovation domain and the open manufacturing domain. The defined workflow focuses on the crowdsourced task derivation

from the design domain to the manufacturing domain. Data that carries that knowledge in the workflow requires storage and sharing services. Furthermore, in the product fulfillment process of the crowdsourced task, the needs for data storage and sharing are lifted and boarded to new levels. This leads to the design of an information service system for the purpose of management which fulfills IaaS for agents in the cyber platform-driven crowdsourced manufacturing system.

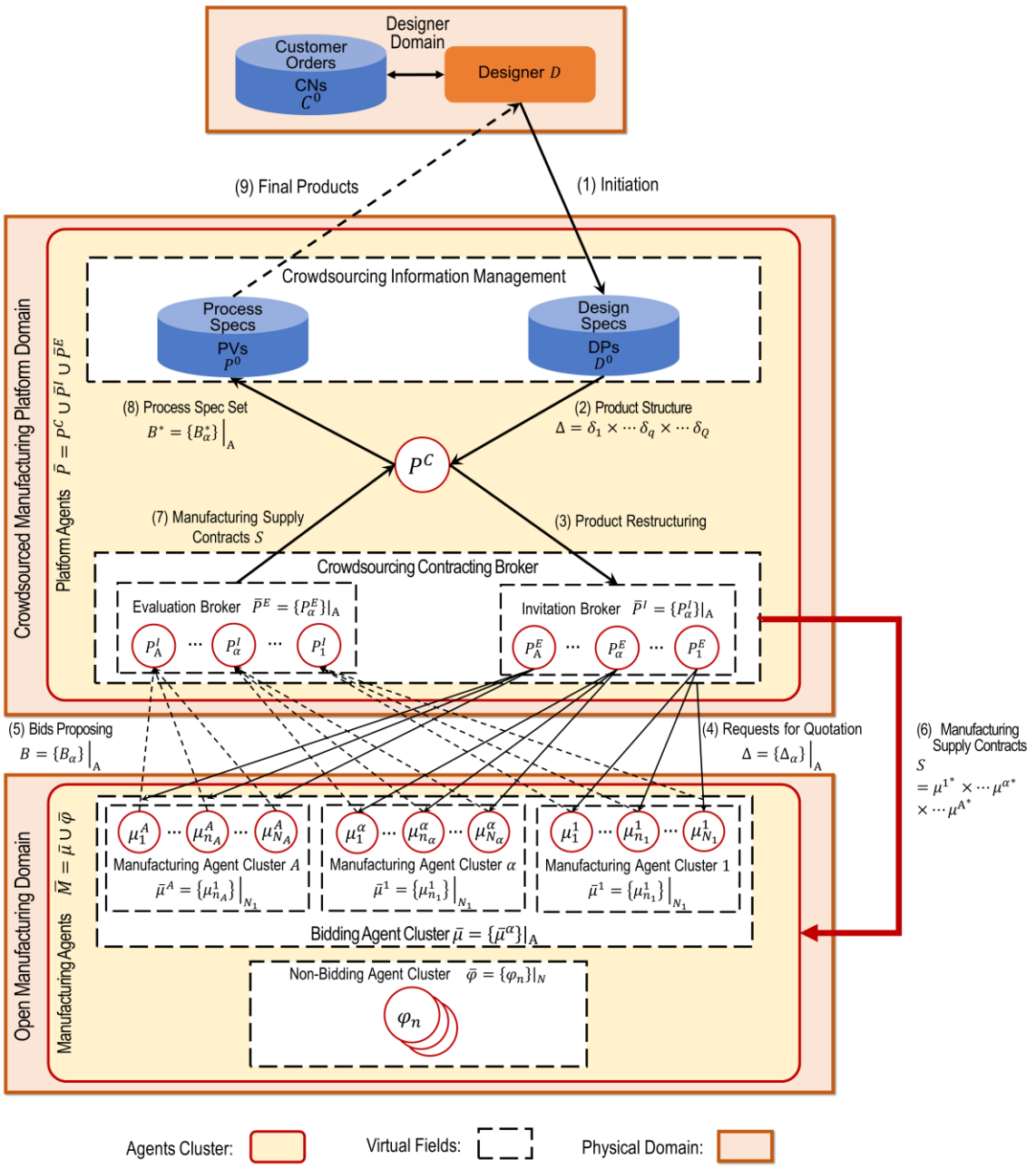


Figure 3-1 Workflow of Crowdsourced Manufacturing (Gong et al., 2022)

3.2 Use Case Analysis of Crowdsourced Manufacturing Cyber Platform

Based on the workflow in section 3.1, a UML case diagram is proposed in this section, as shown in figure 3-2. The UML case diagram reconstructs the workflow in platform-

driven crowdsourcing manufacturing based on the interaction between use cases and users. Three main actors identified for the crowdsourced manufacturing cyber platform are designers, service providers, and managers. Both designers and service providers are crowdsourcers in the crowdsourcing system. The designer is the crowdsourcer that initiates the crowdsourced task. Service providers are crowdsourcers that participate in the fulfillment process of crowdsourced tasks,

The cyber platform is service-oriented. Despite the services provided for third-party users like service providers and designers, it also assists in the coordination of information for managers. The platform serves for contract establishment between desginers and manufacturers, task execution for manufacturers, and management for managers. In the contract establishment process of the crowdsourced task, as indicated in section 3.1, the designers can propose manufacturing tasks on the cyber platform, as shown in figure 3-2. This use case is included in the use case of evaluating proposed tasks that are under the supervision of the manager on the cyber platform. The evaluation of proposed tasks also enables the bids for crowdsourced tasks, which require the participation of service providers. Based on the evaluation of tasks and bidding results, service providers, designers, and managers establish the order contract of crowdsourced tasks through the platform. The contract data is stored on the cyber platform for future reference. Store data is the most significant use case in the crowdsourced manufacturing cyber platform, which includes several sub-use cases. It is the foundation of fulfilling IaaS in services of contract establishment, task execution, and management.

Serving for task execution of crowdsourced tasks, retrieving the real-time task execution or product fulfillment status is necessary for all main actors. To achieve the use

case of retrieving product fulfillment status, the cyber platform should receive and store the product fulfillment data uploaded by service providers first. Furthermore, the process of receiving, storing, and accessing product fulfillment data requires management for the purpose of security and coordination. When uploading the product fulfillment status, the cyber platform inspects the identity of the request sender for security. Similarly, functions like identity verification should be employed for the use case of retrieving product fulfillment status. As mentioned before, storing data is the most significant use case since it participates in all the information sharing and referencing use cases in the platform. Once the product fulfillment process is finished, designers, service providers, and managers evaluate the performance of each agent that provides service in product fulfillment.

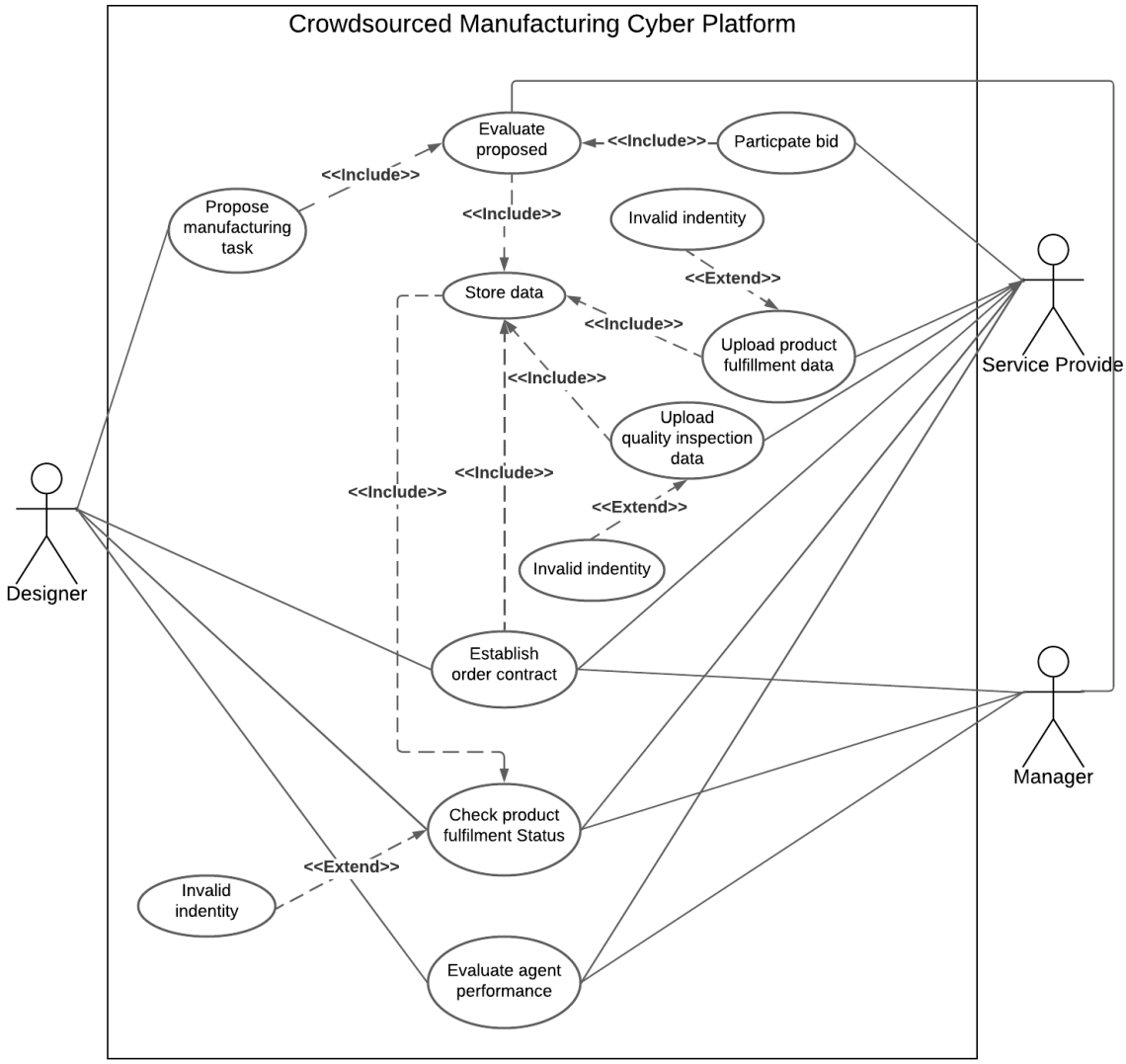


Figure 3-2 UML Case Diagram of Crowdsourced Manufacturing Cyber Platform

Based on the use case and relationship identification of platform-driven crowdsourced manufacturing in the UML case diagram, a unique view can be proposed from the perspective of the fulfillment of IaaS through an information service system that is integrated into the crowdsourced manufacturing cyber platform as shown in figure 3-3. To fulfill the IaaS in task execution and overall information management for crowdsourced

manufacturing, functional requirements in different overlapped areas are identified from the mapping between workflow and use case analysis, as shown in figure 3-3.

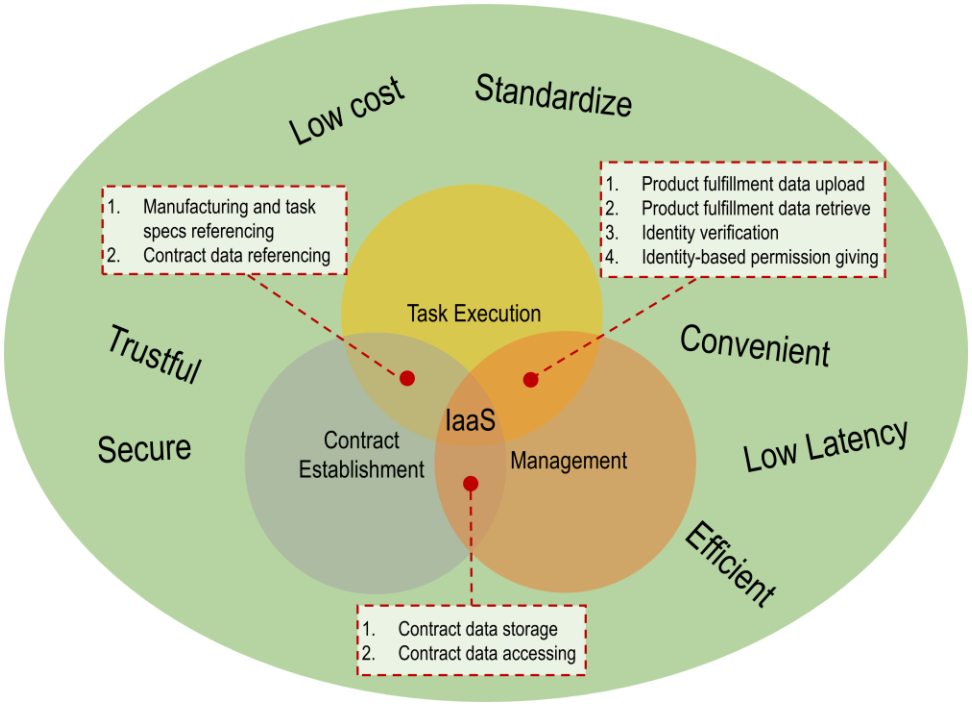


Figure 3-3 Functional Requirements of IaaS Fulfillment System for Cyber Platform-Driven Crowdsourced Manufacturing

In the overlapping between task execution and management areas, functional requirements for an IaaS fulfillment system are identified as (1) product fulfillment data upload, (2) product fulfillment data retrieve, (3) identity verification, and (4) identity-based permission-giving. Management is required in contract establishment for (1) contract data storage and (2) contract data access, as shown in figure 3-3. In the product fulfillment process, users need to access the data stored in the contract establishment process for reference. The IaaS fulfillment system is designed to achieve these requirements with low

construction cost, low time latency, system standardization, high operational conveniences and efficiency, high security, and trust from users.

3.3 Architecture Design of Blockchain-enabled IaaS Fulfillment System

To meet the functional requirements identified in section 3.2, blockchain technology is introduced for the design of the system. Blockchain technology has been widely applied in areas like distributed databases, public ledgers for transactions, and digital events with high security provided by distributed consensus (Crosby et al., 2016). With the aid of smart contracts, automated scripts run in blockchain, new types of synchronized interactions between users and blockchain are enabled in the distributed network without a third party's supervision (Zou et al., 2019). The integration of blockchain ensures traceability, unchangeability, transparency in information sharing and exchanging, and the integration of smart contracts provides a certain level of automation in information management. However, blockchain has limitations in the size and bandwidth, which indicates that higher latency can be caused when the size of sharing data is large (Mending et al., 2018). This brings the question of what kind of file should be posted on the blockchain network, which requires aid from other types of distributed file-sharing systems.

Another term that is brought to the architecture design is Cyber-Physical Systems (CPSs). CPSs digitalize the process in the physical world into computerized entity flows to establish the connection between the bounded physical world and relative boundless cyberspace, which provides services like real-time data access and data processing through the internet (Monostori et al., 2014). Interactions between human to human and human to machine in CPSs have projections on both the actual physical world and digital space. By

integrating CPSs into manufacturing systems based on smart sensing and IoT technologies, material and entity flow in the physical world are digitalized as information flow which leads to interactions between human to human and human to machine. In the system platform-driven crowdsourced manufacturing, the cyber platform is required to allow interaction between crowdsourcers through providing different kinds of services, as mentioned in the previous chpater. This leads to the need to establish a cyber-physical system based on the crowdsourcing platform to coordinate and combine entities in both the physical world and digitalized data. At the same time, fulfillment of IaaS is achieved through this coordination process.

In this section, an overall architecture design is proposed to provide a general view of the proposed system in figure 3-4. Three spaces are included in the figure are information-sharing space, virtual space, and infrastructural space. The virtual space and interactions inside it are achieved through actions and flow in the other two spaces.

Information-sharing space demonstrates the information flow and actions related to data sharing and transferring. Stakeholders in this space are manufacturer and logistics service providers and task managers. Manufacturers and logistics service providers both belong to the term service providers, which is introduced in chapter 3.2. The task manager corresponds to the term manager in chapter 3.2. Service providers upload and retrieve information in the information-sharing space under the supervision of the task manager. Blockchain technologies are employed for constructing a database that contains extracted product fulfillment status. The raw product fulfillment data is stored in the distributed database due to its large size. Contract data, and specifications data are also stored in the distributed database for the same reason. The data stored in the blockchain network is un-

structuralized, which will be structuralized and saved into a database for access. The organization reduces the cost and difficulties of data access management.

Stakeholders in the virtual space are the same as stakeholders in the information-sharing space. In the virtual space, the virtue manufacturing system is constructed based on the established contact of the crowdsourced task. Manufacturer, logistics service provider, and designer who proposes the task participate in the contracting process based on decision support resources. Both crowdsourced task contracts and decision support resources are stored in the distributed database. Product fulfillment status is the real-time task product fulfillment data collected through IoT and smart sensors.

Infrastructure space contains the physical resources for fulfilling the product. The resources and entities in different levels are categorized and configured from the supply chain network. For each logistics service provider, the real-time product fulfillment data is collected and uploaded to their local database, which gives feedback to the local service providers.

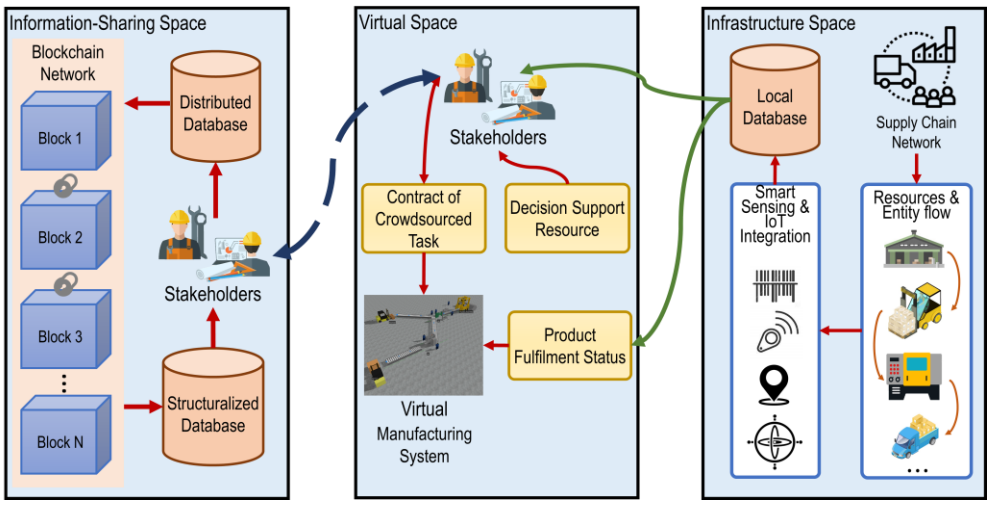


Figure 3-4 Architecture Design of the IaaS Fulfillment System

3.3.1 Information-sharing Space

Figure 3-5 demonstrates details in the information-sharing space. The blockchain network in the information-sharing space store the extracted product fulfillment data. The characteristics of blockchain technologies make the stored data unchangeable and traceable. To initiate the information flow, service providers firstly upload the raw product fulfillment data to distributed database. At the same time, the raw product fulfillment data is extracted to a certain format and sent to the blockchain network through transactions. Blocks are formed through these transactions that contain the extracted data. These three types of transactions in the designed system which are *User_info* transaction, *Manufact_info* transaction, and *Logistics_info* transaction. Three types of transactions represent three types of data stored in the blockchain network.

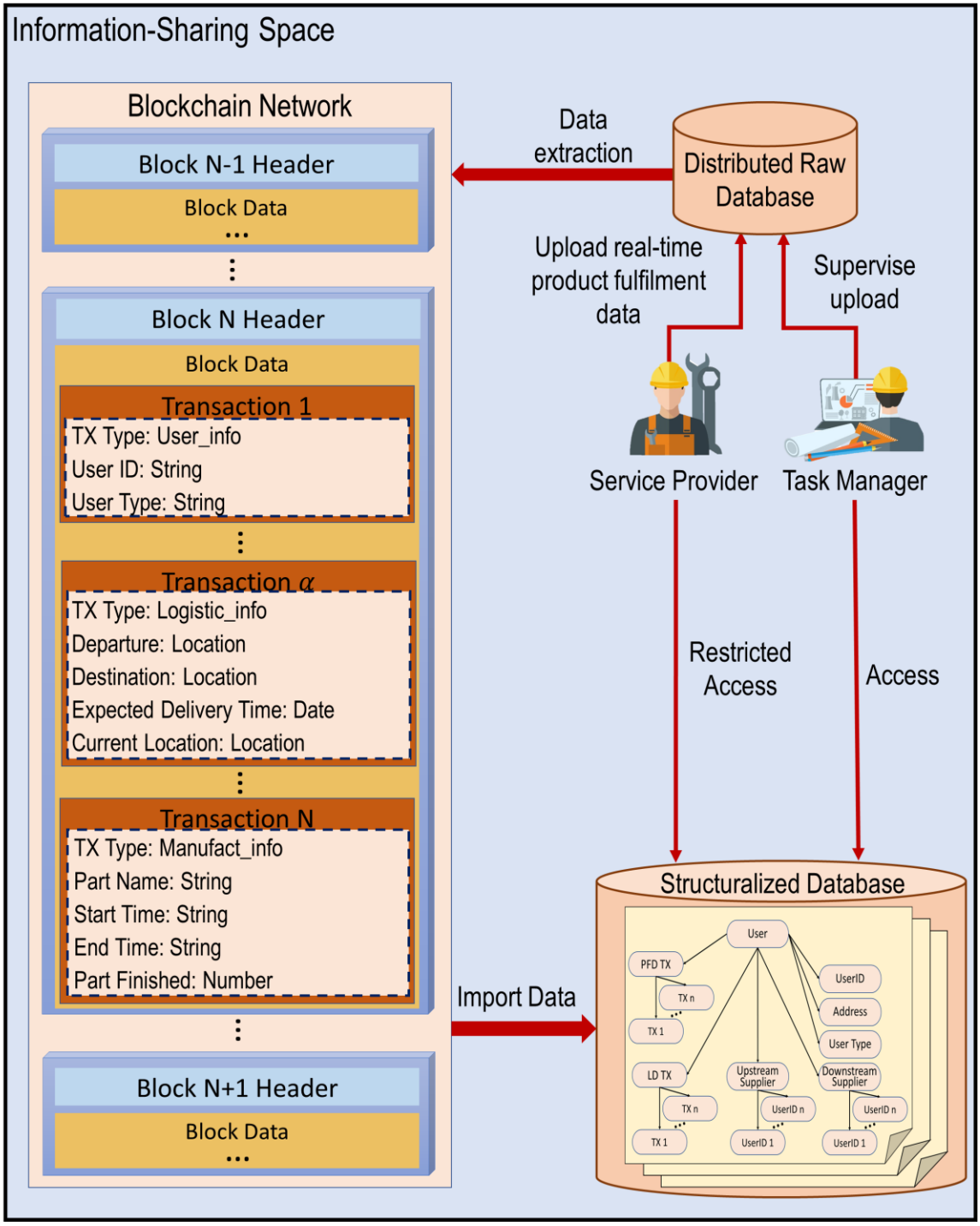


Figure 3-5 Information-Sharing Space

User_info transaction carries user information like user ID and user type. User ID is employed for differentiating different service providers in the crowdsourced task. User

type includes manufacturers and logistics service provider. The hash address is also included in the user information, which is unique for each participant in the blockchain network. The user information is used to provide a reference for identity verification and coordination. Users need to enter their user ID and hash address as passwords to check if they are allowed to perform a certain behavior. Details of the utilization will be discussed in chapter 4. TX_type in the transaction data can label the type of transactions. *Logistics_info* is the second type of transaction which carries extracted logistics data. The logistics data record the physical flow of the entity. *Logistics_info* contains departure location, destination location, current location, departure time, and expected arrival time. This information can describe the status of logistics services. *Logistics_info* transaction is extracted from the logistic data and uploaded by the logistics service provider. The third type of transaction type is *Manufact_info* which contains extracted information on manufacturing data. It contains the name of the part or entity that is manufactured. The manufacturing status is represented by the number of finished parts, the start time of that manufacturing process, and the time when the manufacturing process is ended.

Based on different consensus mechanisms and settings of the blockchain network, the block is mined while containing transactions that carry extracted product fulfillment data. Blocks are mined and connected. Data in the transaction can be retrieved by block ID or hash address of the transactions. The extracted data is stored in the blockchain network discretely where there aren’t connections between these transactions, which causes difficulties in managing and retrieving. Therefore, data in the blockchain is reorganized in struct format and saved to a structuralized database. As shown in figure 3-5, the structuralized data is stored in a distributed database. The structuralized database is updated

when new transactions are sent to the blockchain network. The structuralized data can also be stored into transaction data in a block if the block size is permitted.

The structuralized database provides conveniences for access management. In the general case, the manufacturer and logistics service provider can only access the product fulfillment status of users that are upstream and downstream of the supply chain. Managers of the crowdsourced task can access all users' product fulfillment status in both blockchain and structuralized databases for supervision and coordination.

3.3.2 Virtual Space

The interactions between stakeholders and their behaviors are demonstrated in the virtual space. The decision support resource includes a database of manufacturing specifications and task specifications which is provided during the task proposing process. The task manager, manufacturing service provider, and logistic service provider established the contract from negotiation. During the negotiation process, the duties and obligations of each service provider are clarified and stored in the distributed database. The contract of crowdsourced tasks also includes logistics service specifications, manufacturing service specifications, supply chain structure, and deliverables. Information stored in the crowdsourced contract can establish the structure of the virtue manufacturing model. The virtue manufacturing system digitalizes the layout, precedence in the product fulfillment process.

The manufacturer and logistics service provider upload the real-time manufacturing status and logistics service status, which represents the real-time execution status in the virtue manufacturing model. Interactions and behaviors shown in figure 3-6 are projected

from the physical and virtual processes in the other two spaces, which give a relatively general view of the system.

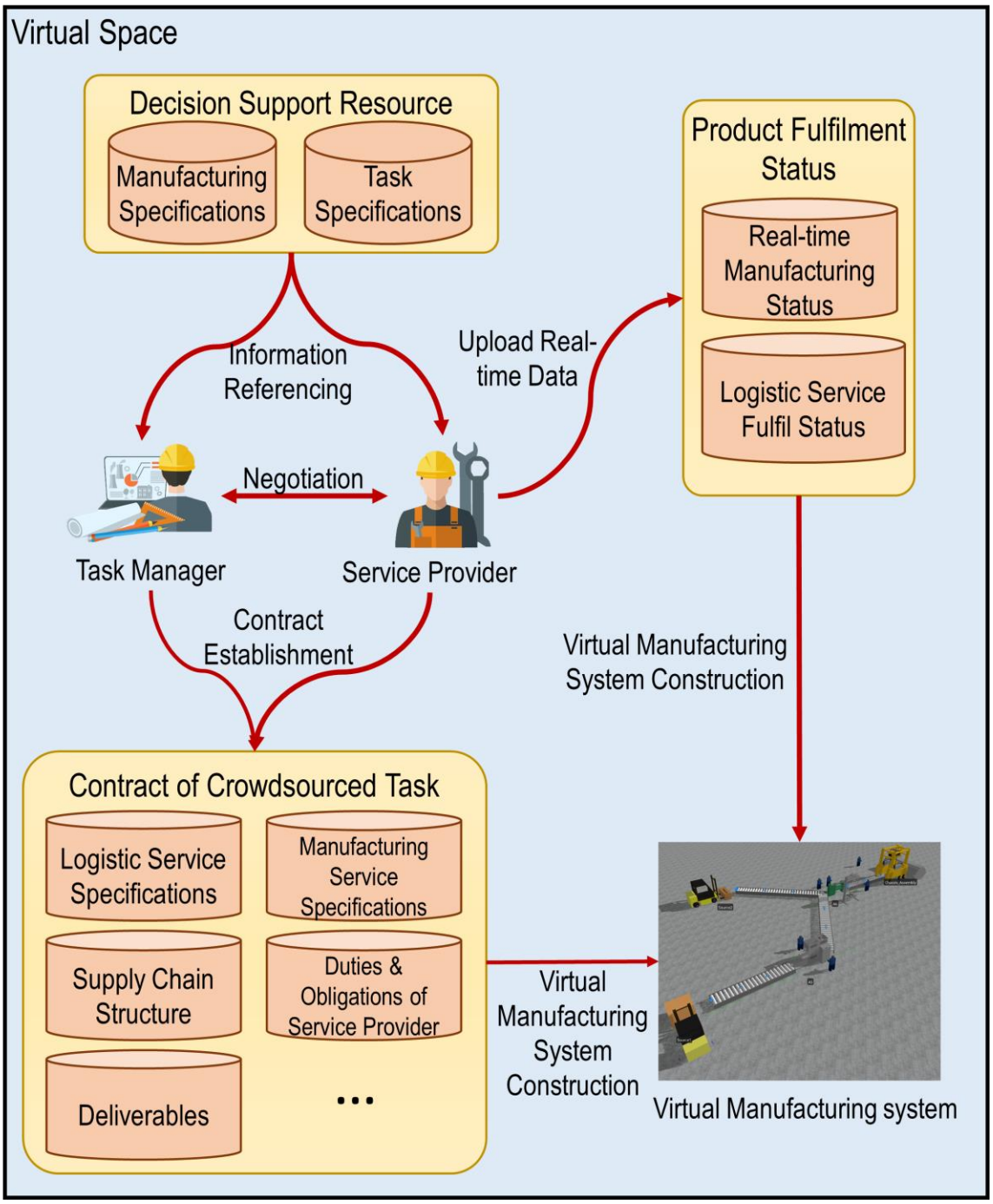


Figure 3-6 Virtual Space

3.3.3 Infrastructure Space

Infrastructure space demonstrates the data acquisition locally for each manufacturing service provider and logistic service provider. The physical resources are configured and categorized based on the raw data uploaded and shared. In the resources layer, changes in inventory level, location changes for forklifts or any other transportations, and numbers of processed parts by machines are digitalized through smart sensing and IoT integration layer through technologies like barcodes, RFID tags, GPS, accelerometers, and other sensors. Like any smart manufacturing system, the data are collected and stored in a local database which gives feedback to the local service provider for analysis and planning.

The task manager can monitor and coordinate the overall product status based on the shared database. The upstream or downstream service providers are also able to acquire some real-time data for production preparations.

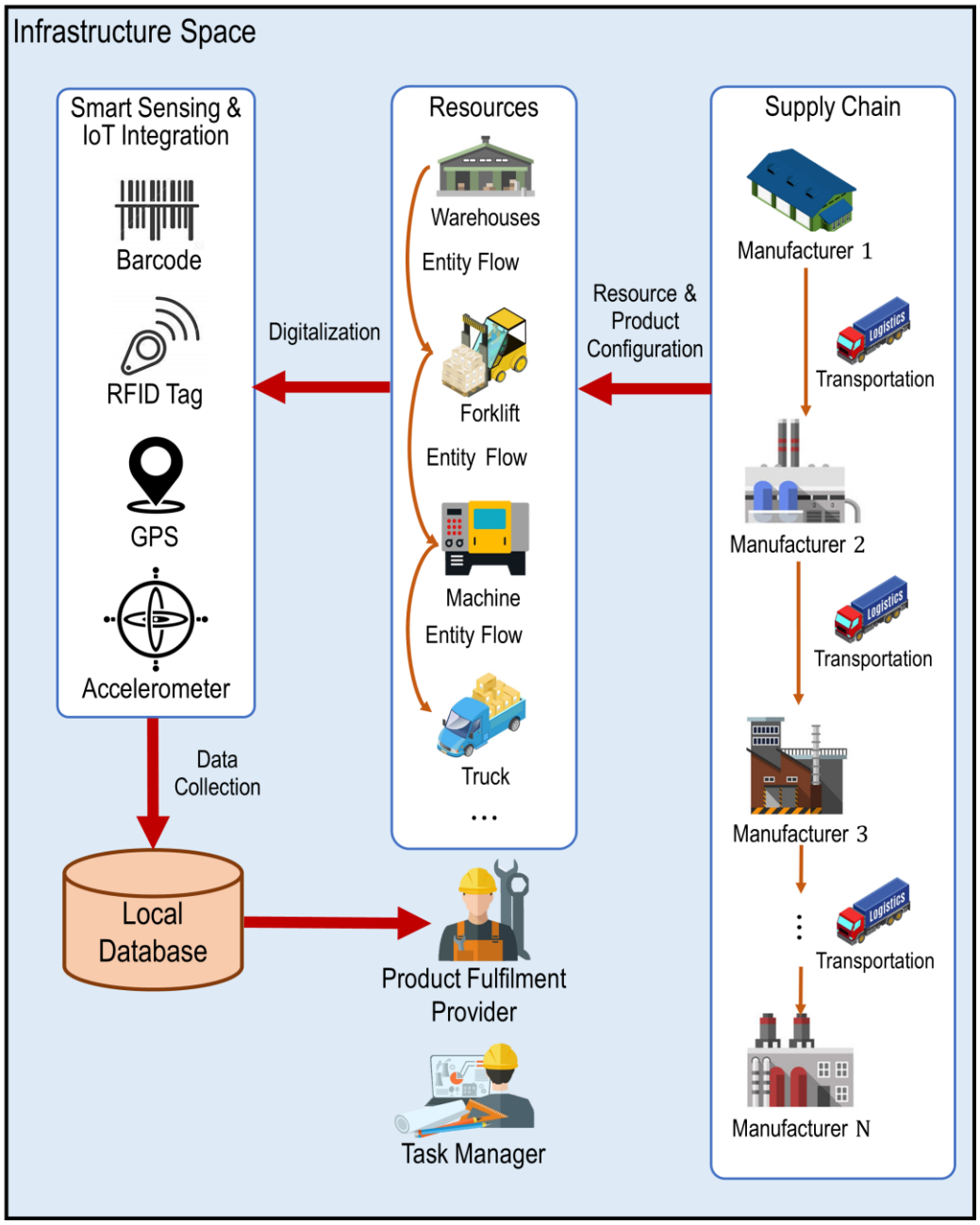


Figure 3-7 Infrastructure Space

3.4 Chapter Summary

In this chapter, the scope is narrowed down from the overall workflow of platform-driven crowdsourced manufacturing to a system architecture design of the proposed system,

which demonstrates the actual interaction. In chapter 3.1, a referenced model of workflow in crowdsourced manufacturing is introduced. In chapter 3.2, a use case diagram is employed to identify the use cases in crowdsourced manufacturing. Focused on use cases for IaaS fulfillment, functional requirements are also identified in chapter 3.2. In chapter 3.3, the architecture design is proposed, which includes perspectives from three different perspectives.

CHAPTER 4. BLOCKCHAIN-ENABLED IAAS FULFILLMENT SYSTEM FOR PRODUCT FULFILLMENT CROWDSOURCING

In this chapter, a unique solution is designed for the IaaS fulfillment system in the product fulfillment crowdsourcing based on the analysis and design discussed in chapter 3. The proposed solution is enabled by blockchain technologies mainly, which allows users to upload, access, manage product fulfillment data easily and securely. This chapter is organized as follows: chapter 4.1 and chapter 4.2 discuss the mechanism of blockchain and other related technologies, and chapter 4.3 introduces key components of the proposed system.

4.1 Blockchain Technologies and Smart Contract

4.1.1 Block and blockchain network structure

Blockchain can be regarded as a digital ledger that records each information and transaction. Each block is added sequentially to the existed block, which records the transaction that happens within a certain time interval. Figure 4-1 shows an example of a blockchain structure. As shown in figure 4-1, the block contains a block header and the main body. The block header consists of the Hash of the previous block, Timestamp, Nonce, and Merkle root:

(1) Hash of the previous block is a 256-bit value that indicates the virtue address of the previous block.

(2) Timestamp is the time when the current block is mined and validated in the blockchain network. Generally, the time is counted since January 1st, 1970.

(3) The nonce is a variance value employed for hash calculation of new block and validation of new blocks.

(4) Merkle root contains all the hash values that point to transactions in the current block.

The main body of the block consists of transaction data. The number of transaction data in a block is affected by the employed block size and each transaction data size. In other words, fewer transactions are recorded in a block if each transaction is larger. Data is encrypted when sending data through the transaction to the blockchain network. Each transaction can be tracked by its hash data which is the transaction hash (TX Hash). The users use a private key to access their account and sign transactions which prove the users' ownership of their address of blockchain network accounts.

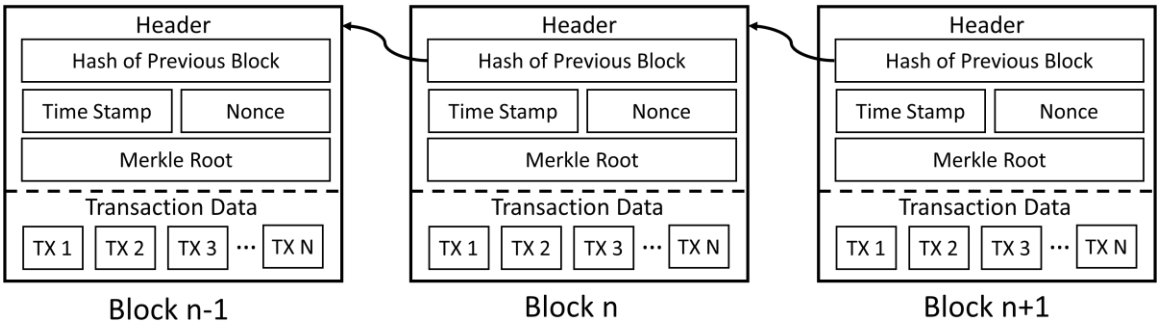


Figure 4-1 Example of Blockchain Structure

Moving from inside to outside of a block, the blockchain network is a sequence block that records a whole list of transactions decentralized with persistency and audibility, which is considered a unique public ledger system (Zheng et al., 2017). In a blockchain network, each block is connected to a previous block except for the first block. The first block which initializes a blockchain network is also called the genesis block.

4.1.2 Consensus mechanism

Trust issues and validation of transactions are solved through consensus mechanisms in the blockchain network safely, efficiently, and conveniently. There are several commonly used consensus algorithms which are proof of work (PoW), proof of stake (PoS), and proof of authority (PoA).

PoW is a non-trust-based consensus mechanism that believes that the node that does the most amount of work has the least probability of attacking the blockchain network through publishing transactions. PoW is employed as the consensus mechanism for the Bitcoin network. The node that reaches a certain level within a value through using the computer to calculate the hash value of the next block can broadcast the new block to other nodes for validation (Nakamoto, 2008). In general, nodes in a PoW-based blockchain network use computational power to solve a question provided by the algorithm, and the first node who solves the problem that meets the algorithm's requirement becomes the miner of that block. PoW wastes lots of resources in calculations of consensus mechanism.

PoS is another non-trust-based consensus mechanism employed by Ethereum (Wood, 2017). Nodes that have more electrical currencies in the blockchain network have less probability of attacking. To avoid the new block being always mined by the richest node, PoS usually combines with other types of algorithms like randomization or new judgment criterion (Zheng et al., 2017). Compared to PoW, PoS wastes fewer resources in the consensus mechanism and has a higher probability of being attacked.

PoA is a trust-based consensus mechanism that is applied to the blockchain network for relatively trustful users. Applications like health data sharing, smart home appliance

management, and distributed control systems employ PoA as a consensus mechanism (Singh et al., 2019, Asad et al., 2020, Yang et al., 2022). A certain number of nodes (usually higher than the half number of total users) are trusted in the blockchain network, and these trusted nodes process the consensus for transactions acted by non-trusted nodes (Angelis et al., 2018).

4.1.3 Smart Contract

The smart contract is developed based on blockchain technology. It is a self-simple executable script based on simple rules and logic. Utilized with masses of pre-defined conditions and rules, the smart contract can respond automatically by executing the pre-defined command to manipulate the blockchain network. The decentralized application (Dapp) is the application that employs smart contracts to provide services to users (Bahga & Madisetti, 2016).

The smart contract has been furtherly developed commonly used in different blockchain platforms with different mechanisms. It has three characteristics which are autonomy, self-sufficiency, and decentralization. As mentioned before, autonomy indicates that the contract is executed automatically after the initialization. Self-sufficiency means that a smart contract can acquire and manage the resources in the blockchain network without the control of the agents. Decentralization is inherited from the characteristics of blockchain technology, where the smart contract is deployed distributivity to all nodes in the network. Furtherly, the smart contract is programmable, which can leverage the possibilities of providing complex services and enclosing the node behaviors in the blockchain network.

4.2 IPFS Distributed File Sharing System

Compared to the blockchain network, Interplanetary File System (IPFS) provides convenience for large file storage with large throughput, which is a distributed file storing and sharing system. Two important characteristics of IPFS are decentralization, content addressing. Decentralization makes it possible for users to access a file not managed by a centralized organization in IPFS. Content addressing means files are identified by their content in the IPFS. Content identifier (CID) labels and points the content in a shared file which is a short cryptographic hash. CID is firstly generated when uploading a file to the IPFS network, which can also be used to access that file by inputting '/ipfs/' + 'CID' in the web browser. A different CID is produced when there is a change in that shared file which indicates both the new version and the old version of the file are stored in IPFS with different CIDs. In other words, the change in the uploaded file is trackable in IPFS.

4.3 Mechanism of Blockchain-enabled IaaS Fulfillment System

Figure 4-2 shows the mechanism of the blockchain-enabled IaaS fulfillment system. The figure shows mechanisms behind the architecture design figure proposed in chapter 3. There are three types of users, as shown in the figure, which are the task manager, the manufacturer, and the logistics service provider. Three types of users interact with IPFS and the blockchain network services provided by a web-based interface. To establish the connection between the web-based interface to the blockchain network and IPFS, smart contracts deployed to the blockchain network, algorithms executed on the web-based interface, and APIs collaborate to process the information and requests. There are two kinds of portals that provide different kinds of services in the web-based interface.

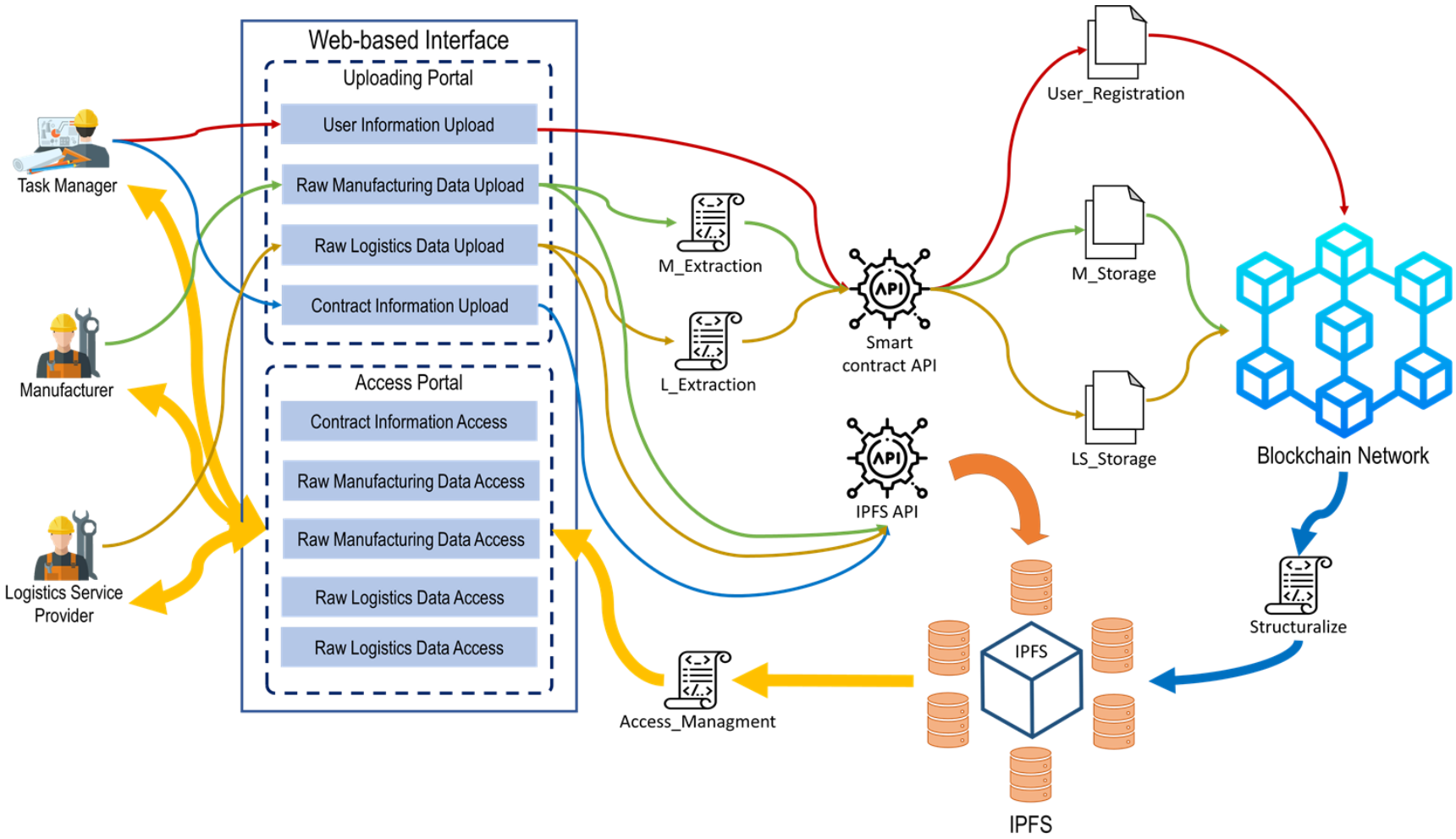


Figure 4-2 Mechanism of the Blockchain-enabled IaaS Fulfillment System

Four services are provided in the upload portal:

(1) **User information upload** allows the task manager to upload user information of participants to the blockchain network. Through a smart contract API, smart contract *User_Registration* is deployed to the blockchain network for sending transaction data. The smart contract only allows the task manager which is the blockchain network initializer to upload the user information. Table 4-1 shows the pseudo code of *User_Registration* smart contract. As shown in the table below, the smart contract firstly checks task manger's identity by checking if its account hash address is the blockchain network initializer. Once the task manager passed the identity inspection, it can input the user id, username, user type, and user's account hash address and the smart contract will store those data to the blockchain network. The user id, username and users' account address hash are strings. User type is defined as integer where "0" represents the manufacturer and "1" represents the logistics service provider.

Table 4-1 Pseudocode of "User_Registration"

Input: *1. User ID, User type, Username and Hash address.*
Output: *Transaction data that contains user information.*
1: **Begin Smart Contract** 2: **create string** *_UserID, _Usernanme, _hash*; 3: **create int** *_Usertype;* 4: **create address** *_hash;* 5: **function** *store*(UserID, Usertype, Username, hash)**:** 6: **If** (AccountAddress==sender.address): 7: *_UserID* = UserID; 8: *_Usertype* =_Usertype; 9: *_Username* = Username; 10: *_hash* =hash;

Table 4-1 Continued

```
11:       Else:
12:          print(‘Access Denied’)
13:    End function
14: End Smart Contract
```

(2) **Raw manufacturing data upload** is the second service in the uploading portal. Manufacturers upload the raw data through the web-based interface, and the web interface will firstly transfer the file to IPFS through IPFS API. After that, the CID of the uploaded file is returned to the web-based interface, and the uploaded raw data file is extracted based on algorithm M_extraction as shown in table 4-2.

Table 4-2 Pseudocode of M_extraction

Input: *Array that records when does a process happen in the system. The process is defined as: an entity occupies a resource for executing a certain action and one entity requires several processes for leaving the system.*

Output: *Extracted data that summarize the overall action exaction results.*

```
1:  Begin
2:    Read array
2:       Acquire the time when the first process happens in the array as t_start;
3:       Acquire the time when the last process ends in the array as t_end;
5:        Acquire the number of entities that are transferred out from the last
          process o in the array as N
6:    return t_start;
7:    return t_end;
8:    return N;
9:  End
```

The extraction algorithm outputs the number of finished tasks or parts from the raw manufacturing data and returns them to the web-based interface. Extracted

manufacturing data and CID of the raw data uploaded in the IPFS are sent to the blockchain network through smart contract *M_Storage*. The pseudo-code of *M_Storage* is shown table 4-3. As shown in table 4-3, the *M_storage* requires four kinds of input which are (i) the CID of the uploaded raw manufacturing data, (ii) the task name inputted by the manufacturer, (iii) the extracted manufacturing data, and (iv) the list that contains account address hash of all manufacturers which is uploaded by the task manager through user information upload function. The *M_storage* sends the transaction data to the blockchain network that contains extracted manufacturing data and the CID for accessing related raw data. The transaction type is also recorded in the blockchain, where the integer "0" indicates that the transaction is related to manufacturing status. The on-chain and off-chain data storage design improves the efficiency of the blockchain by reducing the sizes of transactions. The extracted manufacturing data reflect the product fulfillment status with fewer complexities when accessed by other users.

Table 4-3 Pseudocode of M_Storage

Input: *1. CID outputted from IPFS.* *2. Manufacturing task name* *3. t_{start}, t_{end}, and N outputted from M_extraction algorithm* *4. List of account address hash of manufacturers uploaded by the task manager*
Output: *Transaction data that contains manufacturing information.*
1: **Begin Smart Contract** 2: **Import** account address hash list **as** L 3: **create string** *_t_start, _t_end, _CID, _TaskName _*; 4: **create int** *_N, _type;* 5: **function** *store*(t_start, t_end, CID, TaskName)**:** 6: **If** (sender.address **in** L): 7: *_t_start* = t_start;

Table 4-3 Continued

```
8:          _t_end =_t_end;
9:          _CID = CID;
10:          _TaskName = TaskName;
11:         _type = 0;
12:       Else:
13:          print(‘Access Denied’)
14:    End function
15: End Smart Contract
```

(3) **Raw Logistics Data Upload** is the third service provided in the upload portal for logistics service providers. Similar to the raw manufacturing data upload, the web-based interface firstly uploads the raw logistics data that contains time and location information to the IPFS. During the raw data upload, the web-based interface utilizes the algorithm L_extraction as shown in table 4-4 to obtain key information from the raw logistics data and send the extracted logistics service information, and return CID to the blockchain network through smart contract *L_storage*. The raw logistics data is data that records a list of location information (latitude and longitude) and related time information at those locations, which reflect the location and time changes from the departure location. Pseudocode L_extraction is shown in table 4-4. As shown in the table, L_extraction will output the locations at the departure time and end time from the raw data. In other words, it merges the logistics service conditions in several time steps to the beginning and end conditions in one time step.

Table 4-4 Pseudocode of L_extraction

Input: *Array that records entity's location with time.*
Output: *Extracted information that contains departure location and time current location and time*
1: **Begin** 2: **Read** array 3: *Acquire the time of departure in the array as* t_{depart}*;* 4: *Acquire the location of departure in the array as* C_{depart}*;* 5: *Acquire the time of departure in the array as* $t_{current}$*;* 6: *Acquire the location of departure in the array as* $C_{current}$: 7: **return** t_{depart}*;* 8: **return** $t_{current}$*;* 9: **return** C_{depart}*;* 10: **return** $C_{current}$*;* 11: **End**

The pseudocode of *L_storage* is shown in table 4-5. It requires four kinds of inputs which are the CIDs of the uploaded raw logistics service data, the name of the logistics service task, outputs from algorithm L_extraction, and a list of logistics service providers' account address hash. The deployed smart contract ensures that only the logistics service provider can send this type of transaction to the blockchain network. The transaction is also labeled with integer 1.

Table 4-5 Pseudo-code of *L_Storage*

Input: *1. CID outputted from IPFS.* *2. Logistics service task name.* *3.* t_{depart}*,* C_{depart}*,* $t_{current}$*, and* $C_{current}$ *outputted from L_extraction algorithm.* *4. List of account address hash of logistics service providers uploaded by the task manager*
Output: *Transaction data that contains logistics service information.*

Table 4-5 Continued

```
1:  Begin Smart Contract
2:     Import account address hash list as L
3:   create string _t_depart, _t_current, _C_departure, _C_current, CID
                                _ServiceName;
4:     create int _type；
5:              function store(t_depart, t_current, C_depart, C_current, CID,
ServiceName):
6:          If (sender.address in L):
7:             _t_start = t_start;
8:             _t_current =  t_current;
9:             _CID = CID;
10:            _ServiceName = ServiceName;
11：           _type = 1;
12:         Else:
13:            print(‘Access Denied’)
14:     End function
15:  End Smart Contract
```

(4) **Contact information upload** is the last service in the upload portal. This service is only provided for the task manager for uploading the data in the contract of the crowdsourced task, as mentioned in chapter 3. This data is directly uploaded to the IPFS through the web-based interface. Some CIDs of uploaded files are shared with all the participants in the crowdsourced tasks for accessing files like product specifications for references.

Files in IPFS and data in the blockchain network are updated with the progress of product fulfillment. Meanwhile, the structurization algorithm structuralizes the uncategorized data stored in the transaction data of blocks. The algorithm is designed to be executed at a certain time interval for acquiring the updated data in the blockchain network.

Table 4-6 shows the pseudocode of the structurization algorithm. The algorithm can access the blockchain network directly without any limitations, which is encapsulated from users of the system.

Table 4-6 Pseudocode of Structurization

Input: *1. The blockchain network of the crowdsourced task.*
2. Precedence of the supply chain for the crowdsourced task.

Output: *A list of class instances that contain categorized transaction data and precedence.*

```
1:  Define struct User{
2:      string userID;
3:      string account_hash;
4:      int usertype;
5:      list Manu_TX;
6:      list Log_TX;
7:      list upstream;
8:      list downstream;
9:      function M_TX(transaction_hash) {
10:       appendix transaction hash to the list Manu_TX;
11:     end
12:     function L_TX(transaction_hash) {
13:       appendix transaction hash to the list Log_TX;
14:     end
15:     function addupstream(account_address_hash) {
16:       appendix account address hash to the list upstream;
17:     end
18:     function adddownstream(account_address_hash) {
19:       appendix account address hash to the list upstream;
20:     end
21: End Define
22: Connect to Blockchain network
22: Import Precedence Data
23: Create list Allusers
24: For i in range (1 to number of blocks ):
25:    acquire list of transactions in block i as TX_list;
26:    for each transaction c_tx in TX_list:
27:       acquire string stored in c_tx as extracted_data;
28:       if extracted_data contains user information:
29            acquire userID stored in extracted_data as c_uid;
30:           create a User class instance named with string in c_uid
31:           save sender's account address to c_uid.acount_hash;
```

Table 4-6 Continued

```
32:         save usertype information in c_uid.usertype;
33:         save list of account address hash to list upstream
34:         save list of account address hash to list downstream
35:         appendix the new instance of User class to list Allusers
36:     end
37:     elseif extracted_data contains extracted manufacturing transaction data:
38:         acquire the struct belongs to User class that have same account_hash to the
              sender's
39:         acquire the userID of that struct
40:         global()userID.M_TX(c_tx.address); //append the current transaction hash
                to list Manu_TX of the struct that is named as string contains in
                userID
41          update the instance of User class to list Allusers to replace the existed
                instance
42:     end
43:     elseif extracted_data contains extracted logistics service transaction data:
44:         acquire the struct belongs to User class that have same account_hash to
              the sender's
45:         acquire the userID of that struct
46:         global()userID.L_TX(c_tx.address); //append the current transaction hash
                to list LOG_TX of the struct that is named as string contains in
                userID
47:         update the instance of User class to list Allusers to replace the existed
                instance
48:     end
49: return Allusers
50: End
```

As shown in table 9-6, to structuralize the blockchain data, a class object named "User" is firstly defined. The defined class not only has attributes for storing data but also has class methods for updating attributes. Instances of the "User" class are created and updated through scraping data in the blockchain network. A list of instances of the "User" class is returned and saved into IPFS. In the list, each instance is defined with the user ID and contains different kinds. The structure of "User" class and its instances is shown in figure 4-3. The class has seven attributes. "userID" is a string type attribute for recording

the name of a user. When creating an instance of "User" class, the string saved in "userID" is also used for naming and pointing new instances. "account_hash" is the account address hash in the blockchain network. "usertype" is an integral type of attribute where "0" represents the manufacturer, and "1" represents the logistics service provider. The rest of the attributes are list types where "Manu_TX" contains all the TX Hashes of manufacturing type transactions sent by this user, and "Log_TX" contains all the TX Hashes of logistics service type transactions sent by this user. Therefore, for a manufacturer type of instance, "Log_TX" attribute is empty since the related user is not allowed to update any data or transaction that do not belong to the manufacturing type. "upstream" and "downstream" are lists that contain account addresses hashes of both manufacturers and logistics service providers in the upstream and downstream supply chain.

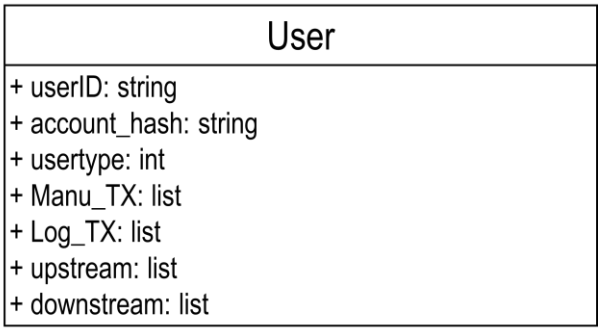


Figure 4-3 Structure of "User" Class

The "structurization" algorithm returns a list that can be considered as a table of content that categorizes all data transacted to the blockchain network. The TX hashes of blockchain transactions are categorized by their senders and types. This list doesn't store the data directly but stores "keys" that can be employed for retrieving those transactions and raw data files. The list is saved as "XML" or "JSON" files in the IPFS.

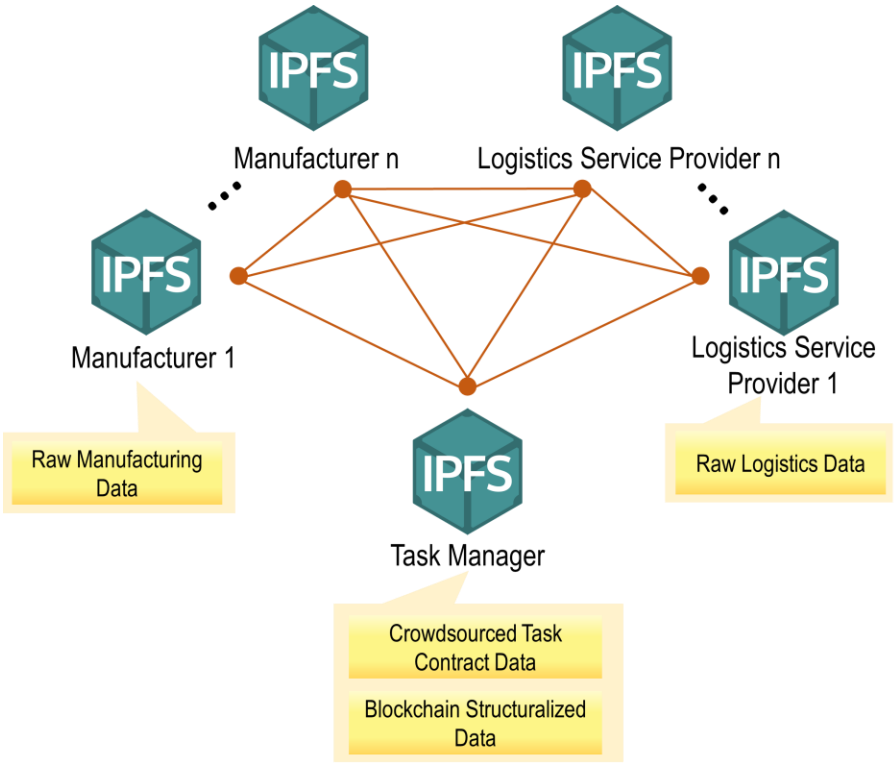


Figure 4-4 IPFS Network in IaaS Fulfillment System

Figure 4-4 shows the IPFS network in the IaaS fulfillment system. As mentioned in figure 4-2, IPFS plays a big role in raw data storage. In such a peer-to-peer (P2P) system, the file is only accessible by CID when the local node switch on the daemon that builds the connection between the local storage to IPFS when other nodes don't host that data. In the proposed system, each node must turn on the daemon locally to ensure that the raw data file is accessible since users don't host data uploaded by other nodes. As shown in figure 4-4, each manufacturer and logistics service provider takes the responsibility of hosting raw data files uploaded by them. For the task manager, it must host both contract data of crowdsourced tasks and the structuralized data from the blockchain. Same as other types of users, the task manager doesn't interact with IPFS directly.

As shown in figure 4-2, users retrieve data through the access portal in the web-based interface. There are five services provided in the access portal, which are:

(1) **Contract information access** is employed for accessing contract data of crowdsourced tasks stored in the IPFS hosted by the task manager. The contract data of the crowdsourced task consists of data for product fulfillment process referencing. When the user sends the request for contract information access, "Access_managment" algorithm will send CIDs of accessible files to that user based on its identity.

(2) **Manufacturing data access** is used for retrieving manufacturing status from the raw data file in the IPFS. Based on users' identity, "Access_managment" algorithm accesses the blockchain structuralized data to find instances of class "User" that are allowed to be accessed. The algorithm retrieves all TX hashes in the list attribute "Manu_TX" and uses those TX hashes for retrieving transaction data in the blockchain network. The web-based interface can display the extracted manufacturing data and CIDs related to the raw data. The user could use CIDs to access the raw manufacturing data.

(3) **Logistics service data access** is similar to manufacturing data access which is used for retrieving logistics service data. Similarly, "Access_managment" algorithm accesses the blockchain structuralized data to find instances of class "User" that are allowed to be accessed. The algorithm retrieves all TX hashes in the list attribute "Log_TX" and uses those TX hashes for retrieving transaction data in the blockchain network. In this regard, the user could access both extracted logistics service status and raw logistics service data.

4.4 Chapter Summary

In this chapter, section 4.1 introduces the blockchain technologies in detail for providing a better understanding of the mechanism. Section 4.2 goes through the mechanism of IPFS, which plays a big role in the proposed system for fulfilling IPFS. Section 4.3 introduces the detailed mechanism and flow of the proposed system. The system utilizes the web-based interface, smart contract, algorithms, IPFS, and blockchain network to provide IaaS to users by uploading and retrieving product fulfillment status conveniently and securely with low trust cost. Pseudo codes are also included in this section for demonstrating algorithms of smart contracts.

CHAPTER 5. APPLICATION OF BLOCKCHIAN-ENABLED IAAS FOR TANK TRAILER CROWDSOURCED MANUFACTURING

In this chapter, the crowdsourced manufacturing task of the tank trailer is employed for demonstrating the performance and feasibility of the proposed IaaS fulfillment system. In this chapter, the development process of the system and the application of the developed system is also included. IaaS is fulfilled through interactions between a web-based interface, blockchain network, smart contract, and IPFS in the system. Chapter 5.1 introduces the description of tank trailer crowdsourcing. Chapter 5.2 introduces the required tools and environments for the development. Chapter 5.3 applies simulated data to the developed system for validation and evaluation.

5.1 Case Description of Tank Trailer Crowdsourcing

There are huge amounts of product varieties in the tank trailer industry. To contain different kinds of chemical fluids, different materials, parts, designs, and manufacturing procedures are required for production. Figure 5-1 presents the case scenario of a crowdsourced task for a tank trailer. Figure 5-1 shows the genetic product and process structure (GPPS) of the tank trailer that is crowdsourced. The manufacturing process can be structuralized into raw material, sub-assembly, part, and end-product. There are nine assembly processes and two manufacturing processes that connect each element in the structure. The GPPS is clustered into five groups, and each group represents the

crowdsourced tasks for a manufacturing crowdsourcer. As shown in the figure, five manufacturers participated in the tank trailer crowdsourced task.

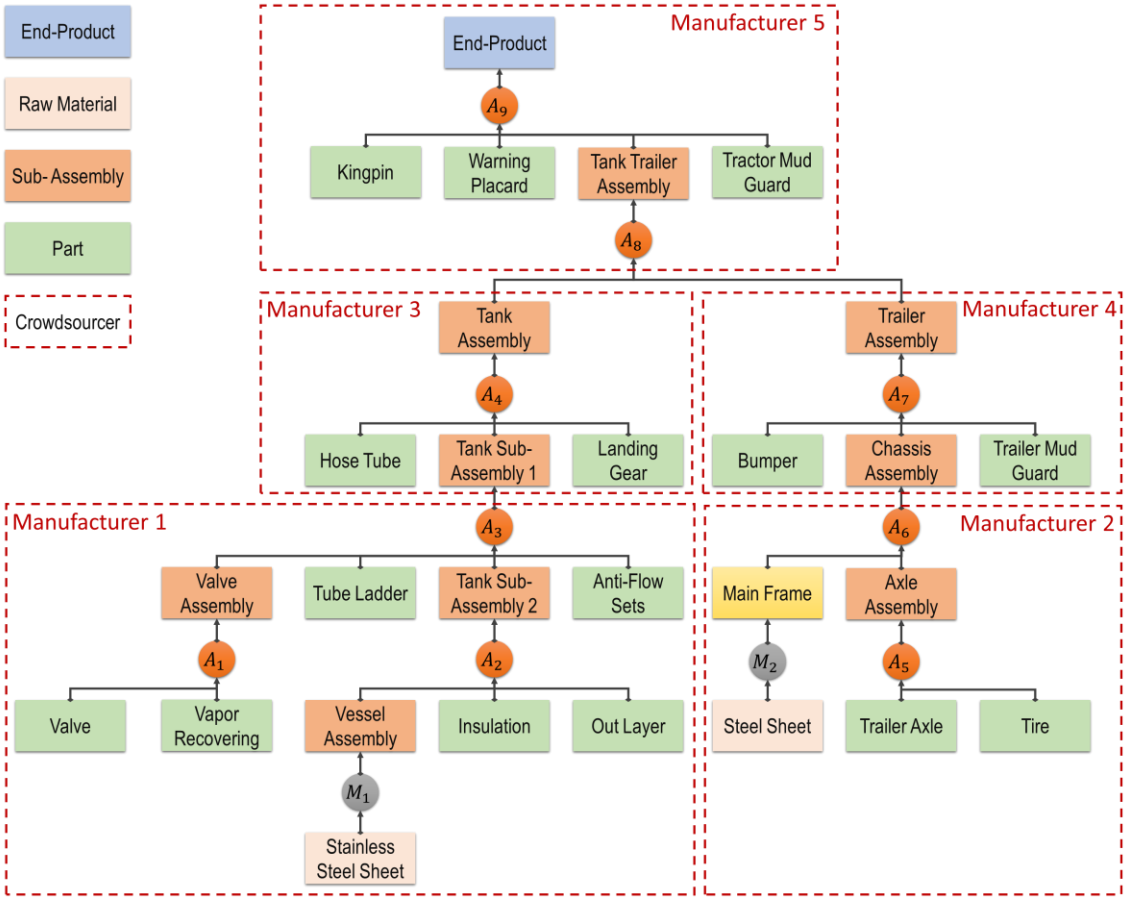


Figure 5-1 Clustered Manufacturing Task for Crowdsourcing

The transportation service is carried out by a logistic service provider who participates in the crowdsourced task. Figure 5-2 demonstrates the case scenario described in the tank trailer crowdsourcing supply chain. Five manufacturers and two logistics service providers who are distributed at different locations geographically participate in the crowdsourced task and try to deliver the final product to the customer. The proposed system

in this case study enables IaaS fulfillment by providing information upload, access service to these participants.

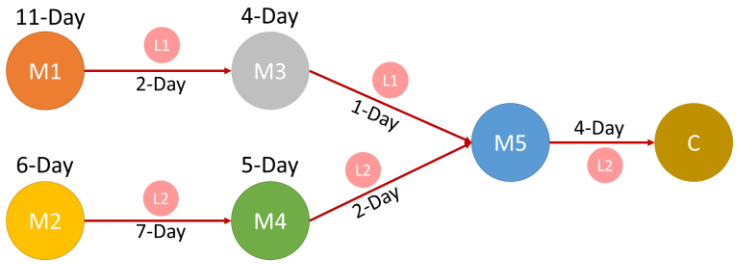


Figure 5-2 Supply Chain of Tank Trailer Crowdsourcing Task

5.2 Developmental Tools and Environments

To meet the functional requirements of the proposed system, the development process of the system in the case study involves several stages using different tools. Table 5-1 shows the tools employed for development and the test environment. The development tools are categorized into five usage purposes which are building a blockchain test network, generating manufacturing data, developing a web-based interface, developing a smart contract, and structuring data stored in the blockchain network.

Table 5-1 Development Tools for the Case Study

Usage	Component	Description
Blockchain test network	Ubuntu Linux 21.10, 8 processors, 32 GB RAM	Test environment
	Ganache-cli	GUI of showing status of the test blockchain network
	Truffle	Test Blockchain network deployment
Manufacturing data generation	Simio	A discrete event simulation software

Table 5-1 Continued

Manufacturing data generation	Simio	A discrete event simulation software
Developing web-based interface	JavaScript, CSS	Programming languages
	Atom	IDE
	Web3.js	Development tool for sending and access transactions to the test blockchain network
	React.js	Development tool for web-based interface
	IPFS	Tool for P2P file sharing
	Meta Mask	Web-based blockchain account management tool
Smart contract development	Solidity	Smart contract developing languages
	Remix	Web-based smart contract deployment tool
Blockchain Structurization	Spyder	IDE
	web3.py	Development tool for sending and access transactions to the test blockchain network

5.2.1 Raw data generation using Simio

To obtain the data for running the experiment on the developed system, the discrete event simulation software Simio is used. There are five Simio models, which represent five manufacturers who participate in the crowdsourced task. The Simio model simulates the manufacturing process of each clustered task and generates the manufacturing status data based on IoT, where each part or sub-assembly is tracked with location and time. Function "mode trace" in Simio can generate the manufacturing data in the format of "CSV". Figure 5-3 shows the example of "model trace data" generated by Simio. This data is generated

from the Simio model of “Manufacturer 5” in the “EntityID” column. The entity “TankTraikerAssembly” is being processed by server “A9”. The data will be extracted when it is uploaded to IPFS through the web-based interface. Figure 5-4 shows the layout of simulation models for five manufacturers built by Simio.

Time(Hou	EntityID	ObjectName	ProcessID	StepName	Action
0	--	--	--	--	System initialization completed.
0	--	--	--	--	Time until next event of timer 'Source1.EntityArrivals' is '0' Hours.
0	--	--	--	--	Firing event 'Source1.EntityArrivals.Event'.
0	--	--	--	--	Time until next event of timer 'Source1.EntityArrivals' is '4.71800197898968' Hours.
0	Source1	Source1	0.OnEntityArrival	[Begin]	Process 'Source1.OnEntityArrival' execution started.
0	Source1	Source1	0.OnEntityArrival	[Create] Entities	Creating '1' new object(s) of entity type 'TankTrailerAssembly'.
0	Source1	Source1	0.OnEntityArrival	[Create] Entities	Created object 'TankTrailerAssembly.11'.
0	TankTrailerAssembly.11	Source1	1.OnEntityArrival	[Transfer] ToProcessing	Entity 'TankTrailerAssembly.11' transferring from '[FreeSpace] Source1' into station 'Source1.Processing'.
0	Source1	Source1	0.OnEntityArrival	[End]	Process 'Source1.OnEntityArrival' execution ended.
0	TankTrailerAssembly.11	Source1	2.OnEnteredProcessing	[Begin]	Process 'Source1.OnEnteredProcessing' execution started.
0	TankTrailerAssembly.11	Source1	2.OnEnteredProcessing	[EndTransfer] IntoProcessing	Entity 'TankTrailerAssembly.11' ending transfer into station 'Source1.Processing'.
0	TankTrailerAssembly.11	Source1	2.OnEnteredProcessing	[Transfer] ToOutputBuffer	Entity 'TankTrailerAssembly.11' transferring from '[Station] Source1.Processing' into station 'Source1.OutputBuffer'.
0	TankTrailerAssembly.11	Source1	1.OnEntityArrival	[End]	Process 'Source1.OnEntityArrival' execution ended.
0	TankTrailerAssembly.11	Source1	0.OnEnteredOutputBuffer	[Begin]	Process 'Source1.OnEnteredOutputBuffer' execution started.
0	TankTrailerAssembly.11	Source1	0.OnEnteredOutputBuffer	[EndTransfer] IntoOutputBuffer	Entity 'TankTrailerAssembly.11' ending transfer into station 'Source1.OutputBuffer'.
0	TankTrailerAssembly.11	Source1	0.OnEnteredOutputBuffer	[Transfer] ToOutputNode	Entity 'TankTrailerAssembly.11' transferring from '[Station] Source1.OutputBuffer' into node 'Output@Source1'.
0	TankTrailerAssembly.11	Output@Source1	1.OnEnteredFromAssociatedObjec	[Begin]	Process 'Output@Source1.OnEnteredFromAssociatedObject' execution started.
0	TankTrailerAssembly.11	Output@Source1	1.OnEnteredFromAssociatedObjec	[Fire] EnteredEvent	Firing event 'Output@Source1.Entered'.
0	TankTrailerAssembly.11	Output@Source1	1.OnEnteredFromAssociatedObjec	[Transfer] ToOutboundLink	Entity 'TankTrailerAssembly.11' transferring from '[Node] Output@Source1' onto link 'Path1'.
0	TankTrailerAssembly.11	Path1	3.OnEntered	[Begin]	Process 'Path1.OnEntered' execution started.
0	TankTrailerAssembly.11	Path1	3.OnEntered	[Assign] EntityMovementRate	Assigning state variable 'TankTrailerAssembly.11.Movement.Rate' the value '5040' Meters per Hour. Previous value was '5040' Meters per Hour.
0	TankTrailerAssembly.11	Path1	3.OnEntered	[EndTransfer] OntoPath	Entity 'TankTrailerAssembly.11' ending transfer onto link 'Path1'.
0	TankTrailerAssembly.11	Path1	3.OnEntered	[End]	Process 'Path1.OnEntered' execution ended.
0	TankTrailerAssembly.11	Source1	2.OnEnteredProcessing	[End]	Process 'Source1.OnEnteredProcessing' execution ended.
0	TankTrailerAssembly.11	Source1	0.OnEnteredOutputBuffer	[End]	Process 'Source1.OnEnteredOutputBuffer' execution ended.
0	TankTrailerAssembly.11	Output@Source1	1.OnEnteredFromAssociatedObjec	[End]	Process 'Output@Source1.OnEnteredFromAssociatedObject' execution ended.
9.92E-05	TankTrailerAssembly.11	Output@Source1	1.OnExited	[Begin]	Process 'Output@Source1.OnExited' execution started.
9.92E-05	TankTrailerAssembly.11	Output@Source1	1.OnExited	[Fire] ExitedEvent	Firing event 'Output@Source1.Exited'.
9.92E-05	TankTrailerAssembly.11	Output@Source1	1.OnExited	[End]	Process 'Output@Source1.OnExited' execution ended.
0.003075	TankTrailerAssembly.11	Path1	1.OnReachedEnd	[Begin]	Process 'Path1.OnReachedEnd' execution started.
0.003075	TankTrailerAssembly.11	Path1	1.OnReachedEnd	[Transfer] FromPath	Entity 'TankTrailerAssembly.11' transferring from '[EndOfLink] Path1' into node 'Input@A9'.
0.003075	TankTrailerAssembly.11	Input@A9	0.OnEnteredToAssociatedObject	[Begin]	Process 'Input@A9.OnEnteredToAssociatedObject' execution started.
0.003075	TankTrailerAssembly.11	Input@A9	0.OnEnteredToAssociatedObject	[Fire] EnteredEvent	Firing event 'Input@A9.Entered'.
0.003075	TankTrailerAssembly.11	Input@A9	0.OnEnteredToAssociatedObject	[Transfer] ToAssociatedObject	Entity 'TankTrailerAssembly.11' transferring from '[Node] Input@A9' into object 'A9'.
0.003075	TankTrailerAssembly.11	Input@A9	0.OnEnteredToAssociatedObject	[Transfer] ToAssociatedObject	Entity 'TankTrailerAssembly.11' transferring from '[Node] Input@A9' into station 'A9.InputBuffer'.

Figure 5-3 Manufacturing Status Data Generated by Simo

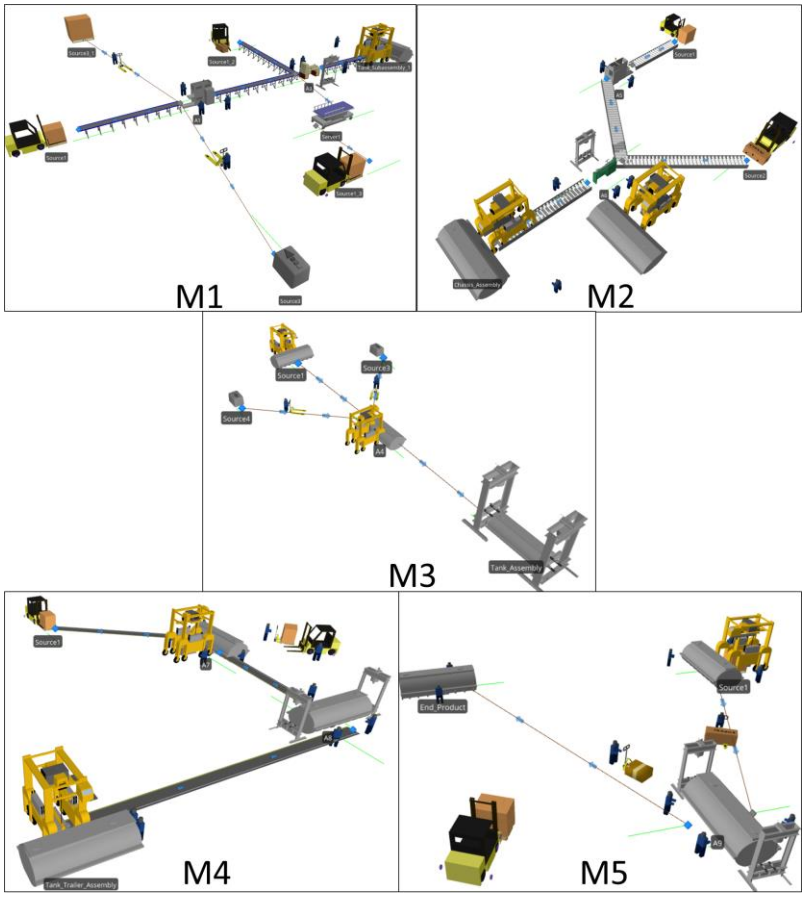


Figure 5-4 Model Layout in Simio

5.2.2 Test network Construction using Truffle & Ganache

The whole case study is applied based on the Ethereum blockchain due to its ability and feasibility in smart contract development support. When testing and deploying the smart contract, a test network is necessary for saving time and cost. Truffle is employed in the case study for building the Ethereum blockchain test network. Properties of the test network like block size, block mining logic, transaction cost can be adjusted by using Truffle. The detailed parameter settings will be introduced in chapter 5.3.

Ganache is a similar tool to Truffle, which is a personal blockchain for developing, deploying, and testing smart contracts. Different from using the command line to interact with the test network, it can show the condition of the test blockchain network through GUI. In this case study, Ganache is a demonstration tool for showing the test blockchain network.

5.2.3 Web-based Interface Development

The development of the web-based interface is the most difficult part of the case study since it must interact with the test blockchain network and IPFS to provide services. In this case study, the web-based interface is developed to achieve data upload, data extraction, sending extracted data to the blockchain. The interface is mainly developed in React.js, which is a JavaScript library for building user interfaces. The Ethereum JavaScript API Web3.js is used for building the connection between the web-based interface and the test blockchain network. The smart contract is deployed using Truffle, and Web3.js, which allows users to use the web-based interface to interact with the deployed smart contract. MetaMask is a web-based blockchain account management tool. Different use cases are achieved by switching Ethereum accounts by using MetaMask.

The smart contract is another essential part of the case study. Solidity is used in the case study for developing smart contracts that are executed based on the Ethereum network. Solidity is an object-oriented and high-level language, and Smart contracts are written in the version of Solidity "^0.6.0" (Solidity, 2022). Smart contracts are deployed by using Truffle, and the web-based interface allows users to interact with the local test blockchain network through API provided by Web3.js.

5.3 Illustrated Working Procedure of the Proposed System

In this chapter, an illustrative case demonstrates the exact procedure of using the developed system. There are five major steps illustrated here which are (1) user information registration, (2) manufacturing status data and logistics service status data upload, (3) raw data extraction, (4) blockchain data structurization, and (5) data retrieving. Different tools are employed for achieving required services at different steps.

5.3.1 Data upload and data extraction

The step 1 to step 3 are included in this sub-chapter. In the case study, the Ethereum blockchain test network is built by using Truffle and exhibited through using Ganache. The local test network is hosted in "127.0.0.1" with port ID "8545". The gas limit used for smart contract deployment is 6721975, and the gas price used for deployment is 20 Gwei. The cost of deploying smart contracts and sending transactions can be ignored since they are related to cryptocurrency transactions which are not considered in the case study. Blocks are mined after a time interval in the blockchain network for user information registration. The time interval is set to be 300 seconds which means all transactions sent to the local Ethereum blockchain network in 300 seconds are contained in the same block. The mining mechanism for manufacturing data upload and logistics service data upload is set to be auto which means one block is mined automatically after sending a new transaction. In this case, one block only contains one transaction related to manufacturing status or logistics service status data.

The first step in using the system is user registration. "URegistration" is the smart contract for registering users' information to the blockchain network. To reduce the

difficulties in development process, the user information in the case study is sent through using Remix. Remix web-based IDE for executing smart contracts, which allows some low-level interaction with the local Ethereum network. Since user registration in this case study doesn't involve the use of IPFS, Remix can be used as the web-based interface for simplifying the development process. In the beginning, the manager of this crowdsourced task takes the responsibility of registering the user information. The manager needs to input the User ID, Username, account hash address, and user type for registering the user to the blockchain network. Figure 5-5 shows an example of user registration. As shown in the figure, the manager inputs the information of the first manufacturer through Remix IDE. The smart contract identifies the account address of the sender to ensure only the task manager can upload the user information.

Figure 5-5 User registration

Step 2 and step 3 require the usage of IPFS. A web-based interface developed by React.js is used in these two steps for data upload. Smart contracts are deployed at different hash addresses in the blockchain network. Those hash addresses are receivers that receive

the different types of transactions from users individually. Smart contracts are deployed after establishing the local Ethereum blockchain network. As shown in figure 5-6, smart contracts are deployed, and their deployed status is shown in the bottom half in the figure. “L_Storage” and “M_Storage” are used for sending transactions that contain logistics services information and manufacturing status information to the blockchain as mentioned in chapter 4.

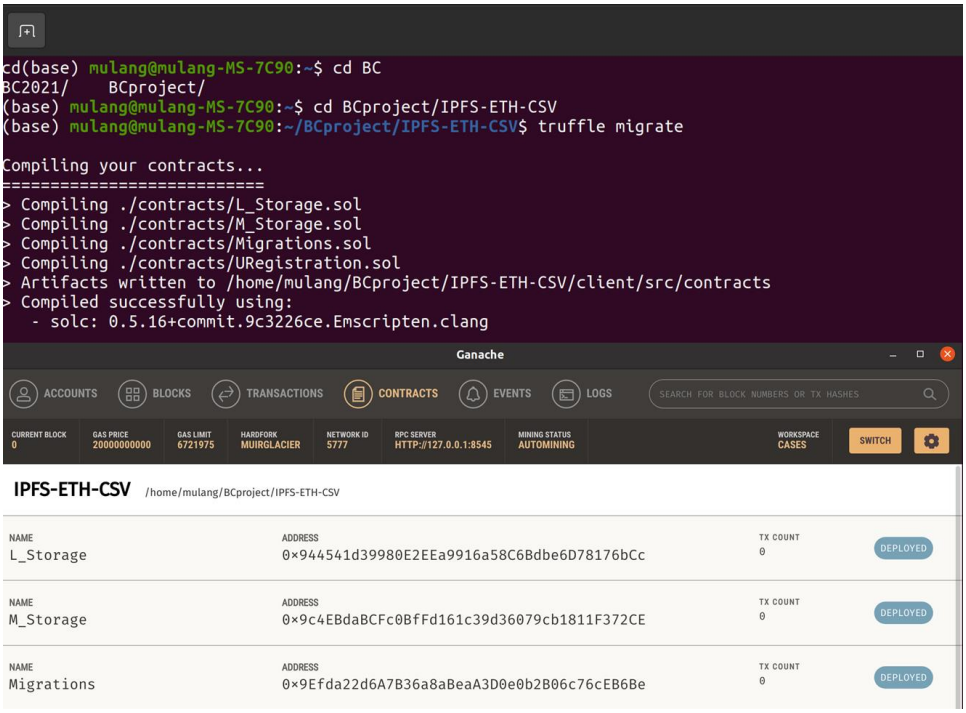


Figure 5-6 Smart Contract Deployment

After the user registration, manufacturers and logistics service providers are supposed to build the connection between their local database to IPFS. Files shared in IPFS are updated after a certain time interval based on the updated local file. At the same time, the raw file shared in IPFS is extracted and sent to the blockchain network through a blockchain API Web3.js. Due to the similarity between manufacturing status data upload

and logistics service status data upload, the use case of the manufacturer is only shown here. Figure 5-7 shows procedures of manufacturing status data upload from "Manufacturer 1". After selecting and uploading the file "M1_Model_trace.csv" by clicking "Return CSV" in the interface, the interface returns the CID of the uploaded file, which indicates the upload is successful. As mentioned earlier, CID is a hash that points to the storage location of the file in IPFS.

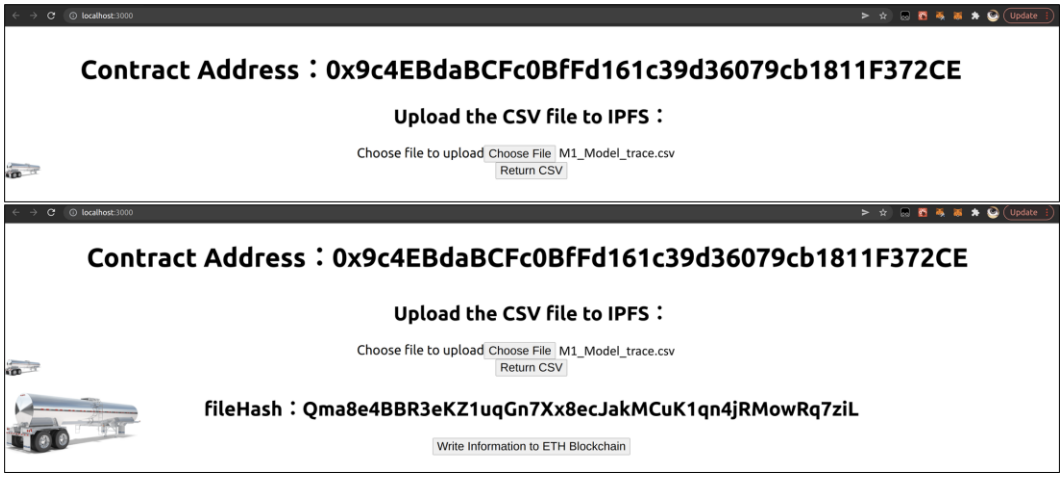


Figure 5-7 Uploading File to IPFS

After sharing the raw data file in IFPS, information in the file can be extracted by clicking "Write Information to ETH Blockchain" button. Figure 5-8 shows the data extraction and transaction process. Once the "Manufacturer 1" clicks the button, the uploaded "CSV" file is read and extracted at the backend of the web-based interface. The data extraction algorithm is based on pseudocode shown in table 4-4. Then, the interface initiates a transaction request of sending the extracted information to the address of the deployed smart contract "M_storage" which is "0x9c4EBdaBCFc0BfFd161c39d36079cb1811F372CE" as shown in figure 5-6 and figure

5-8. Meanwhile, the web-based Ethereum wallet tool MetaMask pops up to let the user confirm the transaction, as shown in the first window in figure 5-8. If the account hash address of the current user matches with manufacturers' account addresses stored in smart contract "M_Storage", the extracted information is transacted to test the blockchain network. As shown in the second window of figure 5-8, the interface returns a message said "Info has been written to blockchain" which indicates a successful blockchain transaction.

Figure 5-8 Data Extraction and Blockchain Transaction

The “manufacturer 1” can check the extracted information by clicking “read info from blockchain” button as shown in the third window in figure 5-8. The interface can print the extracted information. As shown in figure 5-8, it takes 2.28 days for

“Manufacturer 1” to finish manufacturing “tank sub-assembly 1”. The interface also prints the URL for downloading the uploaded file.

During the manufacturing process, each manufacturer and logistic service provider are supposed to upload the data with a similar procedure. Figure 5-9 shows the situation of the test blockchain network. There are 15 blocks mined during the simulated manufacturing process. Block 0 is the genesis block which is mined at the initialization moment of the test network. Block 1 contains seven transactions which represents the registration of five manufacturers and two logistics service providers. Those transactions are sent by the crowdsourced task manager. Block 2 to block 14 contains only one transaction, which is sent by manufacturers and logistic service providers.

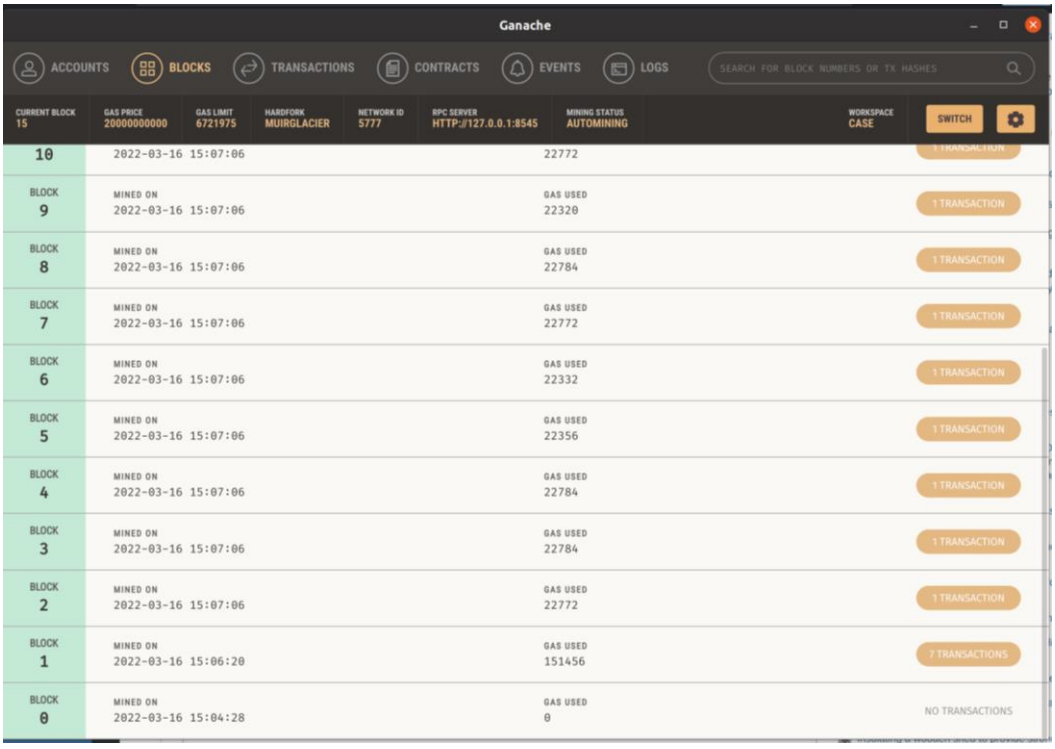
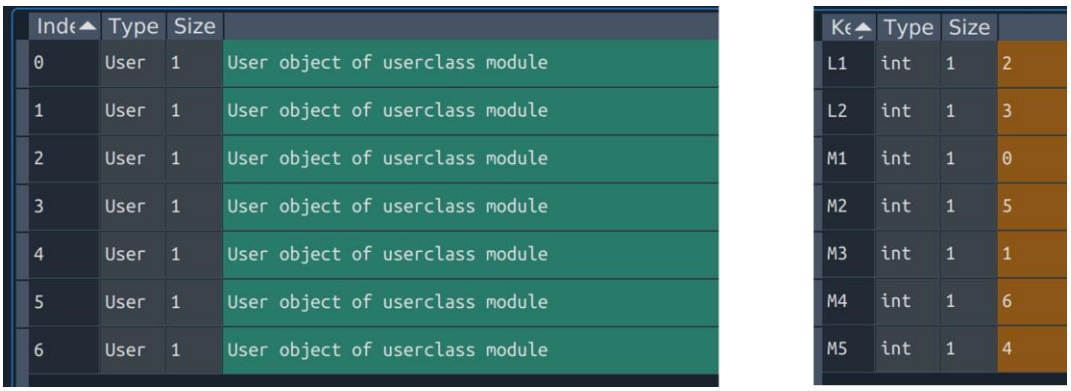


Figure 5-9 Blockchain in the Test Ethereum Network

5.3.2 Block Data Structurization

Structurization and categorization of block data is processed during the whole manufacturing process. It is designed to acquire information in blockchain after a certain time interval. If there are any updates to the blockchain network, the structured block data is updated by storing the new transaction hash address. In this case study, the block data is structured by using Python and Web3.py package. The Python code connects to the blockchain network and obtains information stored in transactions of each block. It first creates several class instances based on user information registered by the task manager. Then transactions are stored in different fields of different class instances based on the senders' addresses. The left side of Figure 5-10 shows the structured block data, which is a list. There are seven instances of "userclass" stored in the list that represent seven crowdsourced task participants. The right side of figure 5-10 shows the username and its index number. For example, "L1" is the username of logistics provider 1. The value "2" represents information of logistic provider 1 is stored in the third element of the list. The structured block data can be serialized and exported, and stored in IPFS or another blockchain network for data retrieving.



Inde	Type	Size	
0	User	1	User object of userclass module
1	User	1	User object of userclass module
2	User	1	User object of userclass module
3	User	1	User object of userclass module
4	User	1	User object of userclass module
5	User	1	User object of userclass module
6	User	1	User object of userclass module

Ke	Type	Size	
L1	int	1	2
L2	int	1	3
M1	int	1	0
M2	int	1	5
M3	int	1	1
M4	int	1	6
M5	int	1	4

Figure 5-10 Structured Block Data

5.3.3 Data Retrieving

Data stored in the block are structured and exported for data retrieving. In this case study, it assumed that the serialized data is stored in another blockchain network instead of IPFS as proposed in chapter 4 due to difficulties in development. Remix is employed again for illustrating the interaction between users and storing structured data in another blockchain network.

The exported structured block data based on figure 5-10 is shown in table 5-2. The serialized and exported structured data is saved to another Ethereum blockchain test network. Table 5-2 is generated based on material in figure 5-10 which the first column in the table represents the index, and the third column represents the username corresponding to the index. For example, “M1” represents manufacturer 1, and its index is “0” according to structured block data shown in figure 5-10. The user type of manufacturer 1 is “1” which represents manufacture. The second column record the account hash address of each user. “Sent Transactions” records the transaction sent by that user. These transaction hashes can be used to access the data carried by them. The data includes extracted information on manufacturing status or logistics service status. It should be noted that CIDs are also included in the extracted information. Therefore, once a user retrieves its accessible transaction hashes, the user can access the extracted information. The user can also access related raw data files by using CIDs that are stored in extracted information. Due to the purpose of demonstration, transaction hashes are simplified to strings in table 5-2. “Upstream User” is the last column that stores indexes of some users in the upstream of the supply chain. The accessibility of data uploaded by other users is controlled by “Upstream User” column. For example, manufacturer 1 and manufacturer 2 don’t have any

upstream users since they initiate the manufacturing process. For manufacturer 3 in the second row, its upstream users are manufacturer 1 and logistic service provider 1 as indicated in the table.

Table 5-2 Simplified Structured Block Data

	Account Hash Address	Userna-me	User Type	Sent Transactions	Upstream User
0	"0x5B38Da6a701c568545dCfcB03FcB875f56beddC4"	"M1"	"1"	"Hash_Tx_M1"	"[]"
1	"0xAb8483F64d9C6d1EcF9b849Ae677dD3315835cb2"	"M3"	"1"	"Hash_Tx_M3"	[0 2]
2	"0x4B20993Bc481177ec7E8f571ceCaE8A9e22C02db"	"L1"	"2"	"Hash_Tx_L1"	[0]
3	"0x78731D3Ca6b7E34aC0F824c42a7cC18A495cabaB"	"L2"	"2"	"Hash_Tx_L2"	[5]
4	"0x617F2E2fD72FD9D5503197092aC168c91465E7f2"	"M5"	"1"	"Hash_Tx_M5"	[1 2 3 6]
5	"0x17F6AD8Ef982297579C203069C1DbfFE4348c372"	"M2"	"1"	"Hash_Tx_M2"	[]
6	"0x5c6B0f7Bf3E7ce046039Bd8FABdfD3f9F5021678"	"M4"	"1"	"Hash_Tx_M4"	[3 5]

In this case study, the simplified structured block data shown in table 5-2 is stored in another test Ethereum blockchain network. Smart contracts are developed which let users access data only uploaded by their upstream users. In the case study, the hash address of the account is employed as a password or identity verification when retrieving data. The smart contract deployed in Remix can only achieve data retrieving if the account exists in the system and has upstream users. Figure 5-11 shows a failed retrieve case. The input account hash address “0x14723A09ACff6D2A60DcdF7aA4AFf308FDDC160C” doesn’t exist in table 5-2. This example shows the security of the data retrieving process.

Figure 5-11 Failed Retrieve Case

Figure 5-12 show a successful retrieving case. Except for the task manager, manufacturer 5 is the only participant that knows its account address. By inputting "0x617F2E2fD72FD9D5503197092aC168c91465E7f2" to the smart contract, it successfully returns user indexes which are "1", "2", "3", and "6". The smart contract also returns the simplified transaction hashes sent by these upstream users. In this example, manufacturer 4 can only retrieve transaction hashes, and it can access the extracted information and raw data files based on those returned transaction hashes.

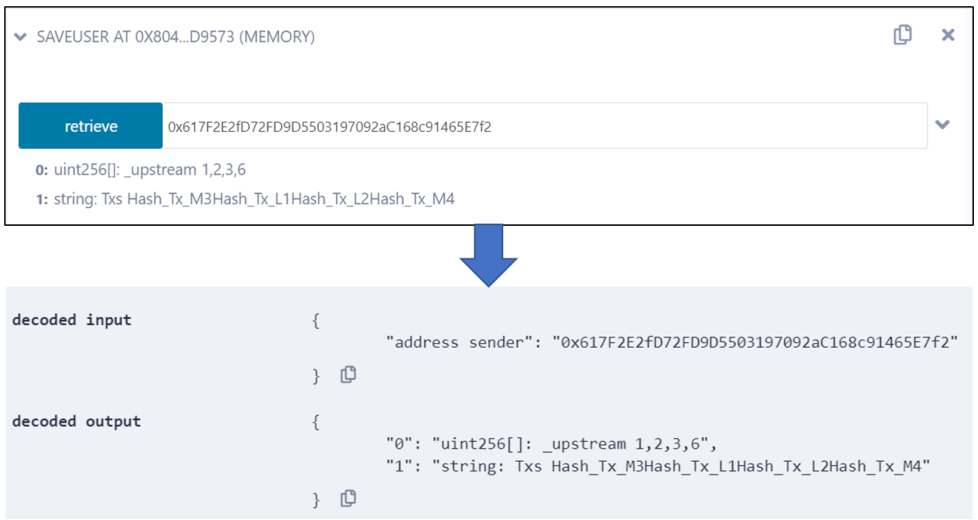


Figure 5-12 Successful Retrieve Case

5.4 Chapter Summary

In this chapter, the proposed system is developed and applied to a case study to test its feasibility. In the case study, the manufacturing of a tank trailer is crowdsourced to five manufacturers and two logistics service providers. Different tools are employed in different developmental stages of the system in different environments. The whole system is developed to work on the test Ethereum blockchain network established by Truffle. Five steps are illustrated in the case study. User information is registered by the task manager by using smart contracts deployed on Remix. Manufacturing status data and logistics service status data are simulated based on Simo. Users use a web-based interface to upload the data to IPFS. The web-based interface extracts the information from raw data and sends transactions to the test blockchain network. If the user's account hash address is verified and the request of sending the transaction is confirmed, the transaction is successful. Block data stored in transactions are structured and categorized during the manufacturing and transportation process. The structured data is imported to another Ethereum blockchain test network for data retrieving. Users use their account hash addresses to retrieve data through smart contracts deployed on Remix. Smart contracts can return accessible transaction hashes, which can be employed for viewing related extracted information and raw data file about the crowdsourced task.

CHAPTER 6. CROWDSOURCING SYSTEM MODELING AND FULFILLMENT CAPACITY BALANCING OPMIZATION

Collaboration is not the only relationship that exists between crowdsourcers. In a cyber platform-driven crowdsourced manufacturing system, manufacturers with different manufacturing capabilities are clustered into different groups. For manufacturers in the same cluster, they tend to participate in similar crowdsourced. Therefore, a competitive relationship exists between manufacturers in the same cluster. As described by the workflow of platform-driven crowdsourced manufacturing in chapter 2.1, manufacturers need to bid for a crowdsourced task. However, the participation level of bidding is not always maximized. When there are too many competitors, the lower possibility of winning the bid and higher bid cost stagnate the increase in the number of bids participated manufacturers. Furthermore, the success of a crowdsourced task requires participation in all decomposed tasks. The differences between scales of manufacturing clusters affect the number of tasks that can be crowdsourced which eventually affects the participation level in bidding. The fulfillment capacity discussed in this chapter is defined as the total manufacturing capacities of active bidding participants in a manufacturing cluster.

This brings a huge management problem to the crowdsourcing platform, and it is hard for decision-makers to come up with plans without understanding the interrelationship between crowdsourcers. Therefore, a model, a simulation method, and an optimization plan are necessary, which benefit the decision-making process for facilitating the platform. A crowdsourcing model is proposed based on the number of participants and difficulties of the crowdsourced task (Guazzini et al., 2015). The model provided a measurement-based

analysis method of a crowdsourcing platform. The population dynamics model based on evolutionary cooperation-competition (ECC) game theory is also proposed (Gong et al., 2021). The model demonstrates the population dynamics interactions between two manufacturing clusters. However, the model assumes that the number of each cluster is infinite. The goal of this chapter is to propose a multi-cluster population dynamics model, a simulation model, and an optimization method. The population dynamics model is based on previous work done by Gong, which provides an analytical solution. However, the population dynamics model assumes that A simulation model is also proposed based on the ECC game theory using the Moran process. The model simulates the interactions between crowdsourcers in a finite group. The proposed optimization method is based on the population dynamics model and simulation results which can assist the platform management.

6.1 Multi-cluster Population Dynamics Model based on ECC Game Theory

In this chapter, the multiplayer-player ECC game model is introduced first to demonstrate the population dynamics of the crowdsourcing platform based on a previous model (Gong et al., 2021). Assuming there are n clusters of manufacturers $c_1, c_2, ..., c_n$. The manufacturers are clustered based on their different types of manufacturing abilities. Manufacturers can bid for a crowdsourcing task that is decomposed into n parts. Therefore, there is only one winner in each cluster who can participate in the crowdsourcing task. This collaboration-competition relationship is modeled by the game theory with several assumptions. As shown in figure 6-1, manufacturers in the same manufacturing cluster c directly interact with each other. In an online platform, every two manufacturers can interact with each other no matter their physical locations or their participation states which

is the first assumption. The second assumption is that the number of agents in each cluster is assumed to be static to reduce the complexity where the total number of crowdsourcers doesn't change with time. The third assumption is that each agent belongs to only one cluster. It is assumed that a manufacturer only has one type of manufacturing capability. The fourth assumption is that each manufacturer in the cluster, it can only choose to bid or not to bid. Therefore, each agent in the model only has two strategies. The fifth assumption is the population in each cluster is finite and large enough. The sixth assumption is that the payoffs of two strategies of manufacturers in one cluster are the same, and they only choose different strategies based on the payoffs. The last assumption is that manufacturers only interact with each other in the same cluster. The effect from the outer cluster is only applied to the payoffs. These assumptions reduce the complexities of modeling. The evolution of the system happens when a manufacturer changes its participation state due to its judgment of the obtainable benefit from bidding. The judgment is modeled based on the payoff matrix in game theory.

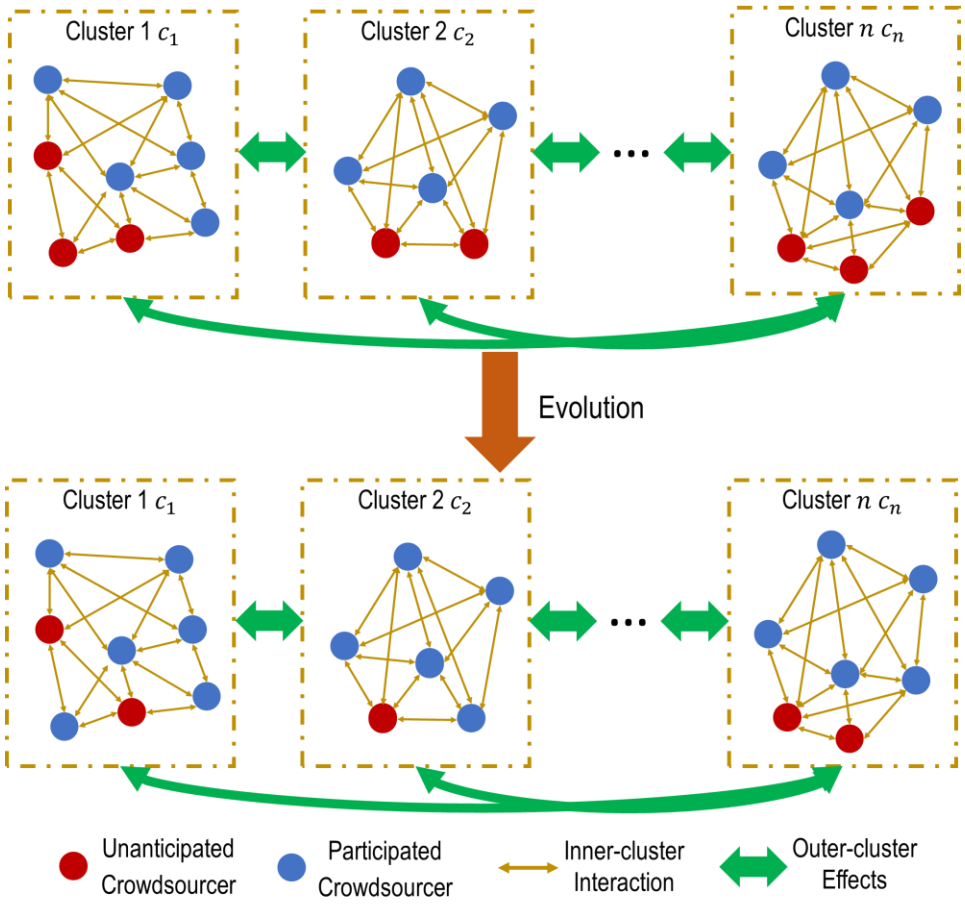


Figure 6-1 Evolutionary Game in Participation of Multi-Clusters

The participation level or fraction of manufacturers that choose to bid in each cluster is defined as $F_1(t),\ F_2(t), \dots, F_i(t), \dots, F_n(t)$. $F_i(t)$ is defined as the number of participating manufacturers over the number of total manufacturers in a cluster. The whole model is constructed in the domain where $0 < F_i(t) < 1, i \in [1, n]$. To construct the payoff matrix based on game theory, the benefit of participating in bidding should be defined. When a manufacturer finishes a certain type of manufacturing task, it receives a profit as a reward. Based on experience in manufacturing, manufacturers in crowdsourcing platforms have expectations in profits of finishing manufacturing tasks. Both participating manufacturers and non-participated manufacturers in one cluster have the expected amount

of profit. For non-participated manufacturers, it is assumed that the reason they don't place bids is they can take manufacturing tasks outside the crowdsourcing platform. Therefore, they gain manufacturing income without the effects of the crowdsourcing platform. For participated manufacturers, the overall benefit of participating in a crowdsourced manufacturing task consists of three parts which are fundamental manufacturing income, biding cost, and crowdsourcing income of the crowdsourced tasks. The fundamental manufacturing income of the ith manufacturing cluster E_i is defined which is the same for all two types of manufacturers in a cluster. Fulfillment capacity factor η_i is used to describe unbalances among manufacturing clusters which are expressed down below:

$$\eta_i = \frac{F_i(t)}{\prod_{1 \le j \le \mathrm{n}, j \neq i} F_j(t)} \qquad (6-1)$$

As described by equation 6.1, the fulfillment capacity factor η_i is constructed based on the participation level of the current cluster over the product of the participation level of other clusters. This number increases when there is a huger difference between the participation level of the current cluster and other clusters. The fulfillment capacity factor η_i can affect the bidding cost in one cluster since the maximum number of potential successful crowdsourced tasks depends on the cluster that has the lowest number of participating manufacturers. The average bidding cost in ith cluster b_i is represented by the equation down below:

$$b_i = ub_i(1 + \eta_i) \qquad (6-2)$$

The average bidding cost comes from the average uncorrected bidding cost ub_i, the bidding cost increases when the participation level of the current cluster is higher than others.

The crowdsourcing income ρ from participating a crowdsourced task is defined in equation 6.3 as shown below:

$$\rho_i = X_i \mathrm{P} \prod\nolimits_{i=1}^{i=n} F_i(t) \qquad (6-3)$$

The crowdsourcing income obtained from a crowdsourced task is the product of a fraction of ith cluster in the total amount of manufacturers X_i, participation level $F_i(t)$, and uncorrected crowdsourcing income P. The participated manufacturer can gain more benefit from the crowdsourced task if the overall participation level is high. Based on equations defined above, the payoff matrix of this two-strategies multi-cluster game model is shown in table 6-1. The dimension of matrix would be $n \times 2$ for a model that has n manufacturing clusters.

Table 6-1 Payoff Matrix of Multi-cluster Evolutionary Game Model

Cluster	Chosen Strategies	
	Bidding (C)	Non-Bidding (D)
1	$f_1^C = E_1 - b_1 + \rho_1$	$f_1^D = E_1$
2	$f_2^C = E_2 - b_2 + \rho_2$	$f_2^D = E_2$
...	...	...
i	$f_\alpha^C = E_i - b_i + \rho_i$	$f_\alpha^D = E_\alpha$
...	...	...
n	$f_n^C = E_n - b_n + \rho_n$	$f_n^D = E_n$

As shown in table 6-2, the average payoff of choosing to bid in ith cluster is f_i^c and f_i^D represents non-bidding behaviors. These payoffs in the matrix represent the averaged output of behaviors. The system of replicator equations is given in equation 6.3 based on

the average payoffs (Hoffbauer and Sigmund, 1998). The system of replicator equations demonstrates the population dynamics in the crowdsourcing platform with n manufacturing clusters.

$$\begin{cases} r_1(t) = \dfrac{dF_1(t)}{dt} = F_1(t) \cdot \big(1 - F_1(t)\big) \cdot (f_1^C - f_1^D) \\ \cdots \\ r_i(t) = \dfrac{dF_i(t)}{dt} = F_i(t) \cdot \big(1 - F_i(t)\big) \cdot \big(f_i^C - f_i^D\big) \\ \cdots \\ r_n(t) = \dfrac{dF_{\mathrm{n}}(t)}{dt} = F_{\mathrm{n}}(t) \cdot \big(1 - F_{\mathrm{n}}(t)\big) \cdot (f_{\mathrm{n}}^C - f_{\mathrm{n}}^D) \end{cases} \quad (6-3)$$

6.2 Moran Process in Multi-cluster ECC Game Model

In this chapter, the Moran process simulation method for the multi-cluster ECC game model based on the payoff matrix in table 6-1 is proposed. Moran process is one of the most popular dynamics processes used in synchronous updating, which can be used to simulate the stochastic dynamics of the evolutionary game model (Gu et al., 2020). The Moran process becomes useful in simulation in this study by transforming the changing process from continuous to discrete. There are two steps in a general Moran process (Traulsen et al., 2005). The first step is to select one agent for reproduction, and the reproduced agent has the same strategy as its parent's. The second step is to randomly select another agent to be replaced by the reproduced one. The total amount of agents is unchanged in the Moran process. Moran process has been applied to an evolutionary game that involves choosing strategies.

There are several assumptions made for the proposed Moran process simulation:

(1) The contracts can be formed with every agent in the population;

(2) Total numbers of manufacturers in cluster c is static.

(3) One manufacturer only belongs to one cluster.

(4) The manufacturer can only select bidding or non-bidding as their strategies.

(5) The total number of manufacturers is finite.

(6) Manufacturers has same payoffs in one cluster.

(7) Manufacturers in a cluster have the same probability of being selected during the simulation.

These assumptions are similar to the assumptions made in chapter 6.1. The number of manufacturers in ith cluster is defined as N_i. The number of manufacturers that place bides in $i$the cluster is defined as j_i. As shown in equation 6.4, the participation level F_i mentioned in chapter 6.1 is calculated form N_i and j_i.

$$F_i = \frac{j_i}{N_i} \tag{6.4}$$

According to the Moran process, which is a Markov process in a special situation, the transition probability of the number of participating manufacturers in ith cluster from state to state is defined in equation 6.5. $T_{j_i}^{+}$ represents the probability that the number of participating manufacturers in cluster i changes from j_i to $(j_i + 1)$. On the contrary, $T_{j_i}^{-}$ represents the transition probability of changing from j_i to $(j_i - 1)$. T_{j_i} represents the probability that the number j_i remains the same.

$$\begin{cases} T_{j_i}^{+} = \dfrac{j_i f_{C_i}}{j_i f_{C_i} + (N-j) f_{D_i}} \dfrac{(N_i - j_i)}{N_i} \\ T_{j_i}^{-} = \dfrac{j_i f_{D_i}}{j_i f_{C_i} + (N_i - j_i) f_{D_i}} F_i \\ T_{j_i} = 1 - T_{j_i}^{+} - T_{j_i}^{-} \end{cases} \quad (6-5)$$

f_{C_i} and f_{D_i} represents the fitness function of selecting strategy C or D in ith manufacturing cluster, which is derived from average payoffs defined in table 6.1 as shown in equation 6.6 (Traulsen et al., 2008). As shown in equation 6.6, w represents the selection intensity. When $w \ll 1$, the selection is weak. The fitness functions are equal to payoffs if $w = 1$, which means a strong selection. w is assumed to be 1 in the presented model.

$$\begin{cases} f_{C_i} = 1 - w + w \cdot f_i^C \\ f_{D_i} = 1 - w + w \cdot f_i^D \end{cases} \quad (6-6)$$

Equation 6.5 can be expressed in a matrix form. If a manufacturing cluster c_i has N_i manufacturers, a $N_i \times N_i$ matrix can show all possible transition probabilities in that cluster. Figure 6-2 shows matrix Φ, $\phi_{a,b}$ is the transition probabilities from $j_i = a - 1$ to $j_i = b - 1$. According to equation 6.5, $\phi_{1,1} = 1, \phi_{1,2} = 0, \phi_{N_i,N_i-1} = 0$, and $\phi_{N_i,N_i} = 1$. This indicates that the process has two stable situations which are $j_i = 0$ and $j_i = N_i$. If the process converges to either situation, the system becomes stable when no manufacturers or all manufacturers participate in the bidding process.

$$\begin{bmatrix} T_{0_i} & T_{0_i}^{+} & 0 & & 0 & 0 & 0 \\ T_{1_i}^{-} & T_{1_i} & T_{1_i}^{+} & \cdots & 0 & 0 & 0 \\ 0 & T_{2_i}^{-} & T_{2_i} & & 0 & 0 & 0 \\ & \vdots & & \ddots & & \vdots & \\ 0 & 0 & 0 & & T_{(N_i-2)_i} & T_{(N_i-2)_i}^{+} & 0 \\ 0 & 0 & 0 & \cdots & T_{(N_i-1)_i}^{-} & T_{(N_i-1)_i} & T_{(N_i-1)_i}^{+} \\ 0 & 0 & 0 & & 0 & T_{N_{i_i}}^{-} & T_{N_{i_i}} \end{bmatrix}$$

Figure 6-2 Transition Matrix in a Manufacturing Cluster

Based on equation 6.5 and figure 6-2, the probability of j_i changes from a to N_i can be expressed in equation 6.4.

$$t_{a \to N_i} = \frac{1 + \sum_{m=1}^{a-1} \prod_{j_i=1}^{m} \frac{T_{j_i}^{-}}{T_{j_i}^{+}}}{1 + \sum_{m=1}^{N_i-1} \prod_{j_i=1}^{m} \frac{T_{j_i}^{-}}{T_{j_i}^{+}}} \qquad (6-7)$$

The fixation probability is defined as the probability that j_i changes from 1 to N_i. The fixation probability can be derived from equation 6.7, as shown in equation 6.8.

$$p_f^i = t_{1 \to N_i} = \frac{1}{1 + \sum_{m=1}^{N_i-1} \prod_{j_i=1}^{m} \frac{T_{j_i}^{-}}{T_{j_i}^{+}}} \qquad (6-8)$$

When $p_f^i > 0.5$, bidding strategy C can eventually replace the non-bidding strategy D for all manufacturers in c_i under strong selection when $w = 1$. If we assume that the update in

strategy change only happens after a bidding process, all manufacturers will participate in the bidding process after a finite number of bids. However, this requirement won't always be satisfied for all clusters at the same time without manually changing the payoffs in some clusters.

6.3 Optimization Strategy Based on Population Dynamics Model and Moran Process Simulations

In this chapter, an optimization strategy is proposed to analyze the current participation level and provide a possible solution to facilitate the participation level in each cluster. To understand how the participation level can be lifted, further analysis of the multi-cluster population dynamics model is necessary. The phase diagram of a 3-cluster population dynamics model is shown in figure 6-4. Three nullclines in the figure are defined by $r_1 = 0, r_2 = 0$, and $r_3 = 0$. r_1, r_2, and r_3 are the system of replicator equations of the 3-cluster population dynamics model based on equation 6-3. The red dot in the figure is the intersection of three nullclines which is the internal equilibrium point. When a solution exists in the real space for $r_1 = r_2 = r_3 = 0$. The solution is the only internal equilibrium. The other two equilibrium points are at $(0,0,0)$ and $(1,1,1)$. The vector arrows figure are gradients of $F_1(t)$, $F_2(t)$, and $F_3(t)$. Three nullclines separate the whole phase diagram into two parts by the growth rate of a participation level in each cluster. For the area that is above the three nullclines, it leads to the convergence where $F_1(t)$, $F_2(t)$, and $F_3(t)$ are equal to 1. If the initial point in the space is in this area, it will eventually converge to $(1,1,1)$. This is the situation in which all manufacturers in the three clusters are willing

to bid for manufacturing tasks. On the contrary, if the participation level is below three nullclines, the participation level of each cluster will eventually converge to 0.

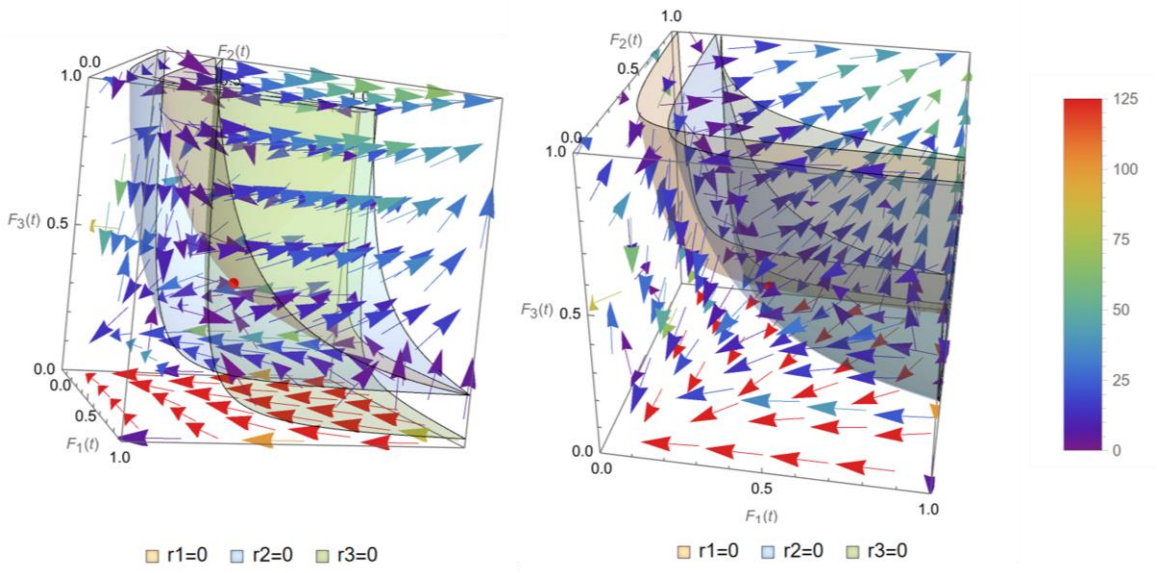


Figure 6-3 Phase Plot of a 3-cluster Population Dynamics Model

For the crowdsourcing platform, it can give subsidies to the manufacturers to reduce the bidding cost b and increase the crowdsourcing income ρ by increasing the uncorrected crowdsourcing income P. By doing this, three nullclines will move in the space, which lets the current participation state stay in the area above the three nullclines rather than below. A general optimization problem can be defined as shown in equation 6.9. The objective function is the length of vector projection of $\vec{u}$ on $\vec{v}$ where $\vec{u}$ is the gradient of $F_1(t), F_2(t), \dots, F_n(t)$ and $\vec{v}$ is the unit vector of direction from $(F_1(t), F_2(t), \dots, F_n(t))$ to (1,1,…,1). The objective function is defined based on two aspects: (1) The direction of the gradient at the current state should point to (1,1,…,1) as much as possible since this is the shortest route for reaching the feasible location. (2) The magnitude of the gradient at the current state is as high as possible. The objective function $f(\mathrm{P}, b_1, b_2, \dots, b_n)$ reflects the

overall growth rate of participation level at the current point. For a n-cluster situation, there are $2n + 1$ sets of constraints. Constraints define the feasible region for each optimizable input like uncorrected crowdsourcing income P and bidding cost b. The last set of constraints ensures the participation level F_i is always above the nullcline $r_i = 0$ in dimension i. The defined optimization problem provides a deterministic solution for finding the best incentive plan for the platform. However, the defined optimization problem is not stochastic since it is defined based on the population dynamics model. There is a possibility that the actual participation level is located below nullclines returned by optimization. To introduce the stochastic process in the defined problem, the Moran process is utilized for simulating the changes in the participation level. In the Moran process, the continuous evolution process is transformed into discrete update steps. For each step, the number of one type of agent can increase by one, decrease by one, or stay at the same value. The possibilities of these transitions are calculated based on the payoff matrix.

$$\max_{\Pi, b_1, b_2, \ldots, b_n} \quad f(\mathrm{P}, b_1, b_2, \ldots, b_n) = \left\| \frac{\vec{u} \cdot \vec{v}}{\|\vec{v}\|^2} \vec{v} \right\|$$

$$\vec{u} = \begin{bmatrix} r_1^{new} \\ r_2^{new} \\ \ldots \\ r_n^{new} \end{bmatrix}, \vec{v} = \frac{\begin{bmatrix} 1 - F_1 \\ 1 - F_2 \\ \ldots \\ 1 - F_n \end{bmatrix}}{\left\| \begin{bmatrix} 1 - F_1 \\ 1 - F_2 \\ \ldots \\ 1 - F_n \end{bmatrix} \right\|}$$

$$r_i^{new} = r_i(X_1, X_2, \ldots, X_n, \mathrm{P}, b_i) \qquad (6-9)$$

$$\textbf{Subject to} \quad \Pi_{\min} \leq \Pi \leq \Pi_{\max}$$

$$\begin{cases} b_{1,min} \leq b_1 \leq b_{1,max} \\ b_{2,min} \leq b_2 \leq b_{2,max} \\ \ldots \\ b_{n,min} \leq b_n \leq b_{n,max} \end{cases}$$

$$\begin{cases} r_1^{new} \geq 0 \\ r_2^{new} \geq 0 \\ \ldots \\ r_n^{new} \geq 0 \end{cases}$$

The population dynamics model proposed in chapter 6.1 and the Moran process introduced in chapter 6.2 are developed from the same payoff matrix. A general optimization strategy is proposed based on the population dynamics model and Moran process as shown in figure 6-4. The strategy is designed to assist with the management of the crowdsourcing platform from the perspective of making incentive plans at a certain confidence level. There are eight steps in the proposed strategy:

(1) The first step is to obtain the participation level of each cluster based on recent biddings.

(2) The second step is to estimate the parameters like bidding cost b_i, uncorrected crowdsourcing income P, and fundamental manufacturing income E_i in each cluster.

(3) The third step is to construct the n-cluster population dynamics model based on estimations and statistical results obtained in the first two steps and equation 6-3.

(4) The fourth step is to predict the possible participation level F_i after a certain time interval or several update steps. Moran process is used to simulate the change in participation level. Randomness in the Moran process could be achieved by methods like Monte Carlo Simulation (MCS). Π simulated results can be obtained by Π runs of simulation.

(5) The fifth step is to find the predictive samples generated in step 4 within the confidence interval.

(6) The sixth step is to derive the acceptable constraints for variables in the objective function defined by equation 6-9. The constraints depend on how much subsidies could be given to manufacturers in each cluster to reduce the bidding cost b_i or increase the uncorrected crowdsourcing income P.

(7) The seventh step is to solve the optimization problem with defined inputs. All simulated predictive results that lie within the confidence interval from step 5 are substituted to r_i^{new}. $r_i^{new} > 0$ is one of the constraints of the optimization problem which ensures that the returned result can benefit all possible outcomes.

(8) If feasible solutions are returned by the problem solver, incentive plan could be made based on the result.

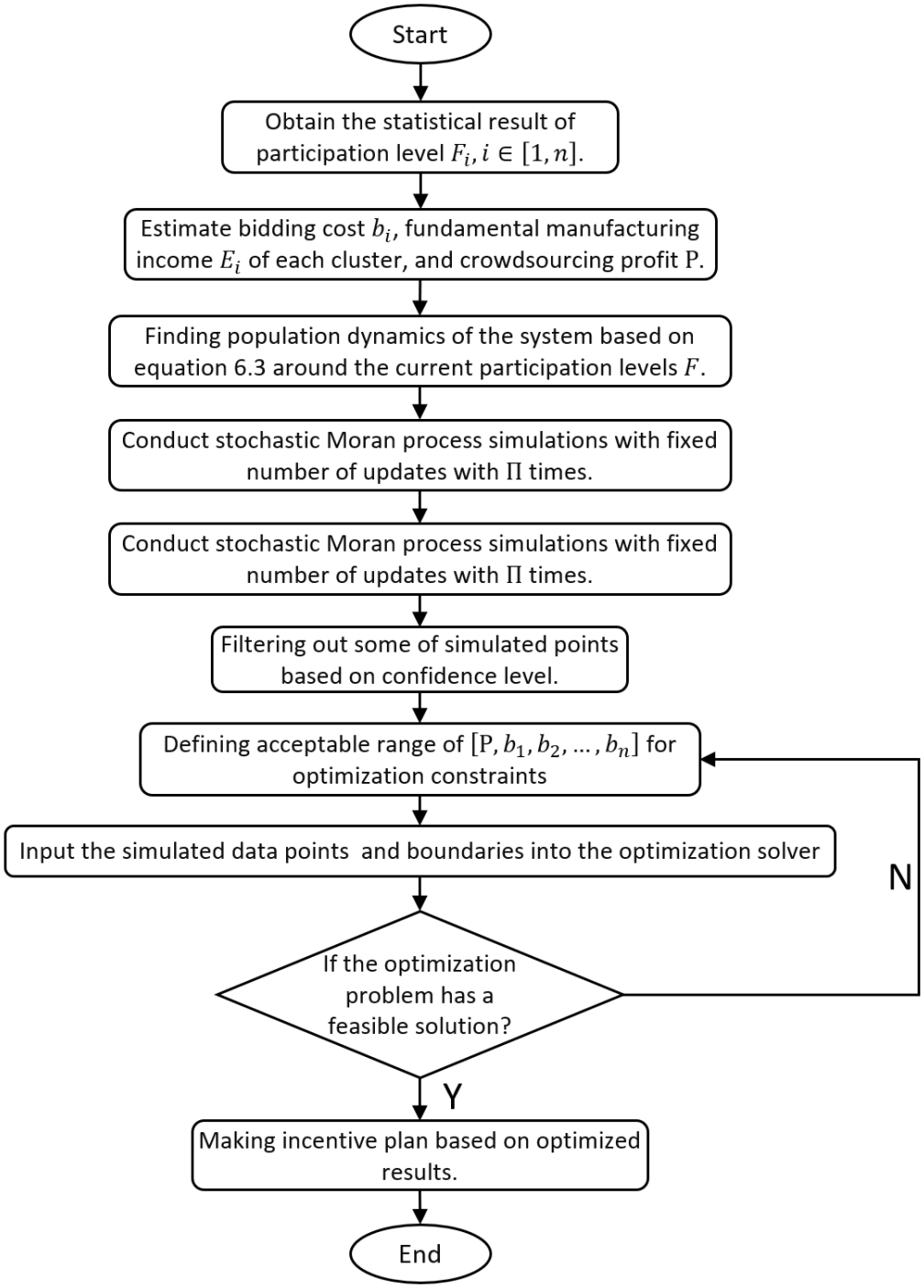


Figure 6-4 Workflow of the Proposed Optimization Strategy

6.4 Example Problem

In this chapter, a test case of the 3-player ECC game model is to illustrate the designed strategy introduced in chapter 6.3. The first step of this example is generating initial state points of $F_1(t)$, $F_2(t)$, and $F_3(t)$. Assuming in a cube with the side length of 1, where the x, y, and z directions represent the $F_1(t)$, $F_2(t)$, and $F_3(t)$ respectively, like showing in figure 6-3. The cube is divided evenly into 1000 small cubes with a side length of 0.1. A random point is selected in each small cube by LHS methods. 1000 times of Moran process simulations are done on each point with 20 update steps. Hence, for each randomly generated point, there will be 1000 predictive points. The parameters of the example problem are summarized in table 6.2. N is the total number of manufacturers. g_1, g_2, and g_3 are fractions for cluster 1, cluster 2 and cluster 3. The number of manufacturers in each cluster is 150, 250, and 100.

Table 6-2 Parameters of Example Problem

Parameter	Mean	Variance
g_1	0.3	N/A
g_2	0.5	N/A
g_3	0.2	N/A
N	500	N/A
Π	100	150
b_1	6	1.5
b_2	10	1
b_3	5	0.75
ρ_1	20	3
ρ_2	20	3
ρ_3	20	3

Only 95% of the generated data (within 2 standard deviations) will be used to run the optimization. This is for filtering out the outliers that are not good representatives for the

simulation. As shown in figure 6-5, 95% of data on the right plot is denser and without any extreme cases. The initial point and constraints of variables of the objective function are summarized in table 6-3. The goal of this simulation is to figure out the performance of the current incentive strategy.

Table 6-3 Boundary Conditions of Example Problem

Parameter	Initial Point	Lower Bound	Upper Bound
P	1000	1000	1500
b_1	6	3	10
b_2	10	5	15
b_3	5	2	10

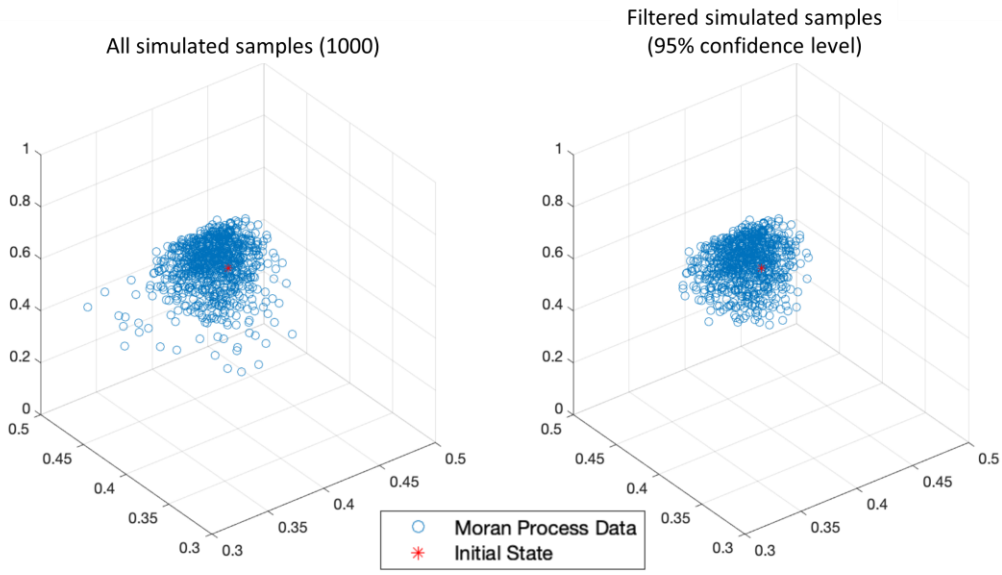


Figure 6-5 Workflow of the Proposed Optimization Method

The experiment shows that under the current optimization strategy, feasible solutions can be returned for 68.8% of randomized participation states, as shown in figure 6-6. From the plot, it can be identified that the infeasible initial states are those that have one or more

participation level that is close to 0. And the points that are in the center region of the cube have the highest growth rate.

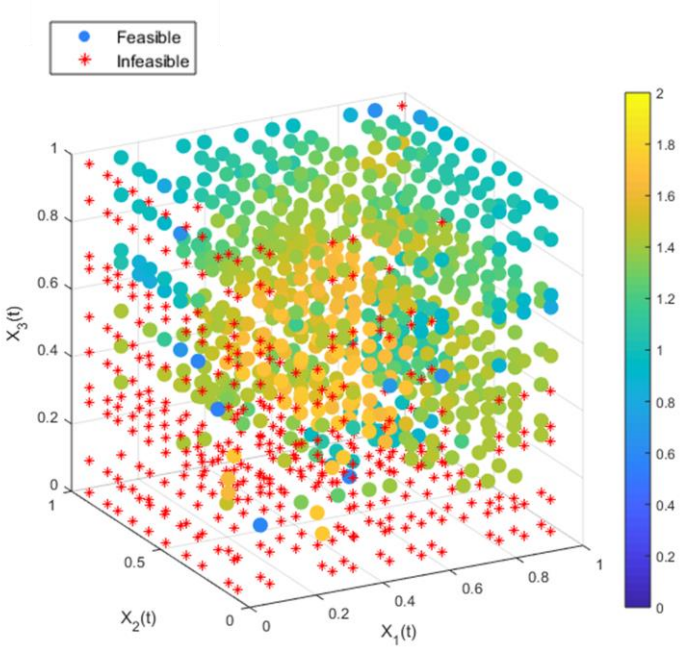


Figure 6-6 Overall Optimization Results for the Different Participation States

6.5 Chapter Summary

In this chapter, the population dynamics model and Moran process of the multi-cluster crowdsourcing platform are introduced. An optimization strategy is proposed to balance the fulfillment capacity of each manufacturing cluster by optimizing the participation level. The strategy is based on both the population dynamics model and the Moran process. An example problem is also included for illustrating the proposed strategy.

CHAPTER 7. CONCLUSIONS

The thesis work proposes two solutions for solving two different problems that exist in cyber platform-based crowdsourced manufacturing. An IaaS fulfillment system for the crowdsourced task execution process is designed based on interactions and information flow in the crowdsourcing platform. A case study is employed for testing the proposed system. An optimal management strategy is also proposed for the fulfillment capacity balancing problem. The proposed strategy can motivate and balance the participation level of different manufacturing clusters in the tournament-based bidding process of crowdsourced work. The optimal management strategy is designed based on the simulation result based on the ECC model, population dynamics, and the Moran process simulation.

7.1 Contributions

The first contribution of this thesis work is identifying use cases and information flow based on the workflow of platform-driven crowdsourced manufacturing. Based on these findings, an IaaS fulfillment system is proposed and developed based on blockchain and IPFS technology. The proposed system fulfills the IaaS by providing information management services. The decentralized system is reliable, which also reduces the cost of trust. The second contribution of the thesis is proposing a population dynamics model and a Moran process model based on ECC game theory to describe the growth rate of a certain type of users in the cyber platform-driven crowdsourced manufacturing system. Furthermore, an optimization strategy is designed based on the population dynamics model and the Moran process simulations to facilitate the growth of users. The platform can manage the crowdsourcing system based on the optimization results.

7.2 Future Work

Future work of this work has three aspects. The first aspect is to apply the proposed IaaS fulfillment system to an actual crowdsourced manufacturing case. With the larger scale of data, the reliability and efficiency of the system can be tested. The second aspect is to develop the proposed system based on the customized blockchain network. Currently, the system is developed based on the Ethereum blockchain network, which limits its development. The third aspect is about the fulfillment capacity balancing. The lack of actual cases makes it hard to validate the model and proposed optimization strategy.

REFERENCES

Albano, M., Sharma, P., Campos, J., & Jantunen, E. (2019). Energy saving by blockchaining maintenance. Journal of Industrial Engineering and Management Science, 2018(1), 63–88. https://doi.org/10.13052/jiems2446-1822.2018.004

Angelis, S.D., Aniello, L., Baldoni, R., Lombardi, F., Margheri, A., & Sassone, V. (2018). PBFT vs Proof-of-Authority: Applying the CAP Theorem to Permissioned Blockchain. ITASEC.

Asad, N. A., Elahi, M. T., Hasan, A. A., & Yousuf, M. A. (2020). Permission-based blockchain with proof of authority for secured Healthcare Data Sharing. 2020 2nd International Conference on Advanced Information and Communication Technology (ICAICT). https://doi.org/10.1109/icaict51780.2020.9333488

Bahga, A., & Madisetti, V. K. (2016). Blockchain platform for Industrial Internet of Things. Journal of Software Engineering and Applications, 09(10), 533–546. https://doi.org/10.4236/jsea.2016.910036

Brettel, M. , Friederichsen, N. , Keller, M. , Rosenberg, M. (2014). 'How Virtualization, Decentralization and Network Building Change the Manufacturing Landscape: An Industry 4.0 Perspective'. World Academy of Science, Engineering and Technology, Open Science Index 85, International Journal of Information and Communication Engineering, 8(1), 37 - 44.

Bücheler, T., & Sieg, J. H. (2011). Understanding science 2.0: Crowdsourcing and open innovation in the scientific method. Procedia Computer Science, 7, 327–329. https://doi.org/10.1016/j.procs.2011.09.014

Christidis, K., & Devetsikiotis, M. (2016). Blockchains and smart contracts for the internet of things. IEEE Access, 4, 2292–2303. https://doi.org/10.1109/access.2016.2566339

Dutta, P., Choi, T.-M., Somani, S., & Butala, R. (2020). Blockchain technology in Supply Chain Operations: Applications, challenges and research opportunities. Transportation Research Part E: Logistics and Transportation Review, 142, 102067. https://doi.org/10.1016/j.tre.2020.102067

ElMaraghy, H., Schuh, G., ElMaraghy, W., Piller, F., Schönsleben, P., Tseng, M., & Bernard, A. (2013). Product variety management. CIRP Annals, 62(2), 629–652. https://doi.org/10.1016/j.cirp.2013.05.007

Gokhale, C. S., & Traulsen, A. (2010). Evolutionary games in the multiverse. Proceedings of the National Academy of Sciences, 107(12), 5500–5504. https://doi.org/10.1073/pnas.0912214107

Gong, X., Jiao, R., Jariwala, A., & Morkos, B. (2021). Crowdsourced manufacturing cyber platform and intelligent cognitive assistants for delivery of manufacturing as a service: fundamental issues and outlook. The International Journal of Advanced Manufacturing Technology, (5-6).

Gu, C., Wang, X., Zhao, J., Ding, R., & He, Q. (2020). Evolutionary game dynamics of Moran process with fuzzy payoffs and its application. *Applied Mathematics and Computation*, *378*, 125227. https://doi.org/10.1016/j.amc.2020.125227

Guazzini, A., Vilone, D., Donati, C., Nardi, A., & Levnajić, Z. (2015). Modeling crowdsourcing as collective problem solving. Scientific Reports, 5(1). https://doi.org/10.1038/srep16557

Hofbauer J, Sigmund K (1998) Evolutionary Games and Population Dynamics (Cambridge Univ Press, Cambridge, UK).

Hosseini, M., Phalp, K., Taylor, J., & Ali, R. (2014). The four pillars of crowdsourcing: A reference model. 2014 IEEE Eighth International Conference on Research Challenges in Information Science (RCIS). https://doi.org/10.1109/rcis.2014.6861072

J. Yang, J. Dai, H. B. Gooi, H. Nguyen and A. Paudel, "A Proof-of-Authority Blockchain Based Distributed Control System for Islanded Microgrids," in IEEE Transactions on Industrial Informatics, doi: 10.1109/TII.2022.3142755.

Jiang, P., & Leng, J. (2017). The configuration of social manufacturing: a social intelligence way toward service-oriented manufacturing. Int. J. Manuf. Res., 12, 4-19.

Jiang, Y., Liu, X., Kang, K., Wang, Z., Zhong, R. Y., & Huang, G. Q. (2021). Blockchain-enabled cyber-physical Smart Modular Integrated Construction. Computers in Industry, 133, 103553. https://doi.org/10.1016/j.compind.2021.103553

Jiao, J., Ma, Q., & Tseng, M. M. (2003). Towards high value-added products and services: Mass customization and beyond. Technovation, 23(10), 809–821. https://doi.org/10.1016/s0166-4972(02)00023-8

Kohler, T. (2015). Crowdsourcing-based business models: How to create and capture value. California Management Review, 57(4), 63–84. https://doi.org/10.1525/cmr.2015.57.4.63

Kusiak, A. (2017). Smart manufacturing. International Journal of Production Research, 56(1-2), 508–517. https://doi.org/10.1080/00207543.2017.1351644

Kusiak, A. (2020). Service manufacturing = process-as-a-service + manufacturing operations-as-a-service. Journal of Intelligent Manufacturing, 31(1), 1–2. https://doi.org/10.1007/s10845-019-01527-3

Li, Z., Wang, W. M., Liu, G., Liu, L., He, J., & Huang, G. Q. (2018). Toward open manufacturing. Industrial Management & Data Systems, 118(1), 303–320. https://doi.org/10.1108/imds-04-2017-0142

Mendling, J., Weber, I., Aalst, W. V., Brocke, J. V., Cabanillas, C., Daniel, F., Debois, S., Ciccio, C. D., Dumas, M., Dustdar, S., Gal, A., García-Bañuelos, L., Governatori, G., Hull, R., Rosa, M. L., Leopold, H., Leymann, F., Recker, J., Reichert, M., … Zhu, L. (2018). Blockchains for Business Process Management - challenges and opportunities. ACM Transactions on Management Information Systems, 9(1), 1–16. https://doi.org/10.1145/3183367

Monostori, L. (2014). Cyber-physical production systems: Roots, expectations and R&D challenges. Procedia CIRP, 17, 9–13. https://doi.org/10.1016/j.procir.2014.03.115

Nakamoto, S. (2009). Bitcoin: A peer-to-peer electronic cash system

Peña, J., Wu, B., Arranz, J., & Traulsen, A. (2016). Evolutionary games of multiplayer cooperation on Graphs. PLOS Computational Biology, 12(8). https://doi.org/10.1371/journal.pcbi.1005059

Perc, M., Gómez-Gardeñes, J., Szolnoki, A., Floría, L. M., & Moreno, Y. (2013). Evolutionary Dynamics of group interactions on structured populations: A Review. Journal of The Royal Society Interface, 10(80), 20120997. https://doi.org/10.1098/rsif.2012.0997

Qu, T., Lei, S. P., Wang, Z. Z., Nie, D. X., Chen, X., & Huang, G. Q. (2015). IOT-based real-time production logistics synchronization system under SMART Cloud Manufacturing. The International Journal of Advanced Manufacturing Technology, 84(1-4), 147–164. https://doi.org/10.1007/s00170-015-7220-1

Sahin, F., & Robinson, E. P. (2004). Information sharing and coordination in make-to-order supply chains. Journal of Operations Management, 23(6), 579–598. https://doi.org/10.1016/j.jom.2004.08.007

Singh, P. K., Singh, R., Nandi, S. K., & Nandi, S. (2019). Managing smart home appliances with proof of authority and Blockchain. Innovations for Community Services, 221–232. https://doi.org/10.1007/978-3-030-22482-0_16

Srai, J. S., Kumar, M., Graham, G., Phillips, W., Tooze, J., Ford, S., Beecher, P., Raj, B., Gregory, M., Tiwari, M. K., Ravi, B., Neely, A., Shankar, R., Charnley, F., & Tiwari, A. (2016). Distributed manufacturing: Scope, challenges and opportunities. International Journal of Production Research, 54(23), 6917–6935. https://doi.org/10.1080/00207543.2016.1192302

Surowiecki, J. (2005). The Wisdom of Crowds: Why the many are smarter than the few and how collective wisdom shapes business, economies, societies, and nations. Anchor Books.

T. Hoßfeld, M. Hirth and P. Tran-Gia, "Modeling of crowdsourcing platforms and granularity of work organization in Future Internet," 2011 23rd International Teletraffic Congress (ITC), 2011, pp. 142-149.

Thuan, N. H., Antunes, P., & Johnstone, D. (2015). Factors influencing the decision to Crowdsource: A systematic literature review. Information Systems Frontiers, 18(1), 47–68. https://doi.org/10.1007/s10796-015-9578-x

Traulsen, A., Claussen, J. C., & Hauert, C. (2005). Coevolutionary Dynamics: From finite to infinite populations. *Physical Review Letters*, *95*(23). https://doi.org/10.1103/physrevlett.95.238701

Traulsen, A., Shoresh, N., & Nowak, M. A. (2008). Analytical results for individual and group selection of any intensity. *Bulletin of Mathematical Biology*, *70*(5), 1410–1424. https://doi.org/10.1007/s11538-008-9305-6

Underwood, S. (2016). Blockchain Beyond Bitcoin. Communications of the ACM, 59(11), 15–17. https://doi.org/10.1145/2994581

Wang, Z., Wang, L., Szolnoki, A., & Perc, M. (2015). Evolutionary games on multilayer networks: A colloquium. The European Physical Journal B, 88(5). https://doi.org/10.1140/epjb/e2015-60270-7

Xu, X., Lu, Q., Liu, Y., Zhu, L., Yao, H., & Vasilakos, A. V. (2019). Designing blockchain-based applications a case study for imported product traceability. Future Generation Computer Systems, 92, 399–406. https://doi.org/10.1016/j.future.2018.10.010

Y. Hao, Y. Li, X. Dong, L. Fang and P. Chen, "Performance Analysis of Consensus Algorithm in Private Blockchain," 2018 IEEE Intelligent Vehicles Symposium (IV), 2018, pp. 280-285, doi: 10.1109/IVS.2018.8500557.

Yang, H., Bao, B., Li, C., Yao, Q., Yu, A., Zhang, J., & Ji, Y. (2022). Blockchain-enabled Tripartite Anonymous identification trusted service provisioning in industrial IOT. IEEE Internet of Things Journal, 9(3), 2419–2431. https://doi.org/10.1109/jiot.2021.3097440

Zhang, L., Luo, Y., Tao, F., Li, B. H., Ren, L., Zhang, X., Guo, H., Cheng, Y., Hu, A., & Liu, Y. (2012). Cloud manufacturing: A new manufacturing paradigm. Enterprise Information Systems, 8(2), 167–187. https://doi.org/10.1080/17517575.2012.683812

Zheng, Z., Xie, S., Dai, H., Chen, X., & Wang, H. (2017). An overview of blockchain technology: Architecture, Consensus, and future trends. 2017 IEEE International Congress on Big Data (BigData Congress). https://doi.org/10.1109/bigdatacongress.2017.85

Zou, W., Lo, D., Kochhar, P. S., Le, X.-B. D., Xia, X., Feng, Y., Chen, Z., & Xu, B. (2021). Smart contract development: Challenges and opportunities. IEEE Transactions on Software Engineering, 47(10), 2084–2106. https://doi.org/10.1109/tse.2019.2942301