KIPS Trans. Comp. and Comm. Sys.
Vol.11, No.6 pp.175~184
ISSN: 2287-5891 (Print), ISSN 2734-049X (Online)

https://doi.org/10.3745/KTCCS.2022.11.6.175

Design and Implementation of an Ethereum-Based Deliverables Management System for Public Information Software Project

Lee Eun Ju† · Kim Jin Wook††

ABSTRACT

Blockchain is being studied in various fields such as logistics, fintech, medical care, and the public sector. In the public information software project, some deliverables are omitted because the developed deliverables and the deliverables requested by the project management methodology do not match, and an additional process is required for payment. In this paper, we propose the deliverables management system for public information software project which is configured a distributed environment using the Ethereum blockchain and which has an automatic payment system only when all deliverables are approved. This system can keep the service available in case of system failure, provide transparency and traceability of deliverables management, and can reduce conflicts between the ordering company and the contractor through automatic payment. In this system, the information of deliverables is stored in the blockchain, and the deliverables that their file name is the hash value calculated by using the version information and the hash value of the previous version deliverable, are stored in the SFTP server. Experimental results show that the hash value of the deliverables registered by the contractor is correct, the file name of the deliverables stored in the SFTP server is the same as the hash value registered in the Ethereum blockchain, and the payment is made automatically to the Ethereum address of the contractor when all deliverables are approved.

Keywords : Ethereum, Smart Contract, Deliverable, Blockchain, Hash

이더리움 기반 공공정보 소프트웨어 사업산출물 관리 시스템 설계 및 구현

이 은 주† · 김 진 욱††

요 약

블록체인은 물류, 핀테크, 의료, 공공 등 다양한 분야에서 연구되고 있다. 공공정보 소프트웨어 사업에서 개발 산출물과 프로젝트 관리 방법론에서 요청하는 산출물이 일치하지 않아 산출물이 누락되는 경우가 발생하고 대금을 지급하기 위해 별도의 프로세스가 필요하다. 본 논문에서는 이더리움 블록체인을 사용하여 분산 환경을 구성하고 모든 산출물이 승인되었을 경우 자동으로 대금이 지급되게 공공정보 소프트웨어 사업산출물 관리 시스템을 제안한다. 이를 통해 시스템 장애가 발생해도 정상 서비스를 제공하고, 산출물 관리의 투명성과 추적성을 제공하며, 자동화된 대금 지급으로 발주사와 수행사 간 갈등을 줄이고자 한다. 본 시스템에서 산출물의 정보는 블록체인에 저장하고, 산출물은 버전 정보와 이전 산출물 파일의 해시값까지 활용하여 계산된 해시값을 파일명으로 SFTP 서버에 저장한다. 실험을 통해 수행사에서 등록한 산출물의 해시값이 정확한지 확인하고, SFTP 서버에 저장된 산출물의 파일명이 이더리움 블록체인에 등록된 해시값과 같은지 확인하였으며, 모든 산출물이 승인되었을 때 수행사의 이더리움 주소로 대금이 자동 송금되는 것을 확인하였다.

키워드 : 이더리움, 스마트 콘트랙트, 사업산출물, 블록체인, 해시

1. 서 론

Satoshi Nakamoto가 2008년에 제안한 블록체인은 Fig. 1처럼 이전 블록의 해시값을 신규 블록의 헤더에 포함하여 체인을 생성하는 P2P(Peer To Peer) 기반의 분산장부이다[1].

블록체인은 전자화폐로 많이 사용되며 물류, 핀테크, 공공, 의료 분야 등에 대한 적용도 연구되고 있다[2]. 물류 분야에서는 SCM(Supply Chain Management) 전 과정에 적용하고 있으며, 핀테크 분야에서는 암호화폐, 기부, 결재 등에 사용되고 있다. 공공 분야에서는 전자투표 및 민원 발급 등에 도입하고 있으며, 의료 분야는 보험금 정산 등에 적용을 준비하고 있다.

공공정보 소프트웨어 개발 사업에서 개발 진행 과정에 생성되는 산출물과 프로젝트 관리 방법론의 산출물이 일치하지 않아 산출물이 누락되는 경우가 발생한다[3]. 발주사는 사업

※ 이 논문은 2021년 한국정보처리학회 ACK 2021에서 "이더리움 기반 공공정보 소프트웨어 사업산출물 관리 시스템 설계"의 제목으로 발표된 논문을 확장한 것임.
† 준 회 원 : 한국전력기술 스마트융합실 선임
†† 정 회 원 : 한국방송통신대학교 컴퓨터과학과 교수
Manuscript Received : December 30, 2021
First Revision : February 23, 2022
Accepted : March 18, 2022
* Corresponding Author : Kim Jin Wook(gnugi@knou.ac.kr)

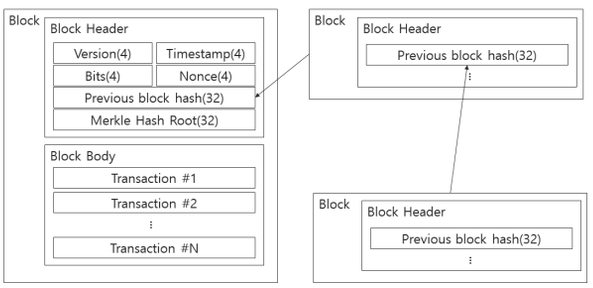

Fig. 1. Construction of Blockchain

종료 시 수행사가 제출한 산출물이 누락 되었는지 확인해야 하며[4], 사업 완료 공문을 접수하여 내부 기안을 통해 대금을 지급하는 프로세스를 수행한다.

소프트웨어 산출물을 관리하기 위한 최근 연구로, 웹 기반 프로젝트 관리 시스템[5]에서 산출물을 공유하는 시스템을 제안하였고, Unity 기반 게임 개발 프로젝트 진행 시 형상 관리 기법[6]에서는 프로그램 소스를 관리하는 기법을 제안하였다. ChainSoft 시스템[7]은 GitHub로 소스 형상을 관리하고 CI (Continuous Integration)에 등록된 테스트 코드를 외주 업체의 프로그램이 통과하면 스마트 콘트랙트에서 코인이 지급되는 시스템을 제안하였고, 분산문서 관리 [8]에서는 전자문서를 IPFS(Inter Planetary File System)에 저장하고, 승인자의 2/3 이상이 승인하면 신규 버전으로 등록되는 시스템을 제안하였다. 이러한 연구들은 산출물을 공유하거나 소스의 형상관리 그리고 전자문서의 버전에 관련된 연구들이다.

단일 서버를 사용한 기존의 시스템들은 장애 발생 시 서비스가 중단된다. 또한 산출물을 모두 제출해도 대금을 지급하기 위해 별도의 절차를 수행해야 한다. 본 논문에서는 이더리움 블록체인을 도입하여 분산 시스템을 구축하여 하나 이상의 노드에 장애가 발생해도 서비스에 문제가 없게 하고 모든 산출물이 승인되면 대금 지급을 자동으로 진행하도록 한다. 또한 이전 산출물의 해시값을 신규 산출물의 해시값 계산에 사용하고 블록에 저장하여 산출물 추적성을 관리한다.

본 논문의 구성은 다음과 같다. 2장에서는 본 논문과 관련한 연구로 발주 프로세스와 스마트 콘트랙트에 대해서 소개한다. 3장에서는 본 논문에서 제안하는 시스템을 설계하고 구현한다. 4장에서는 검증과 분석 결과를 기술하고 5장에서는 결론과 향후 계획을 제시한다.

2. 관련 연구

2.1 공공정보 소프트웨어 사업 단계별 발주 프로세스

소프트웨어 진흥법 제44조 소프트웨어 사업의 과업 범위 3항에 상세 요구사항을 작성하기 위해 단계별 발주를 권고하고 있다[9]. 단계별 발주는 설계와 구현의 2단계로 분리하여 발주하는 방식으로 요구사항 명확화를 위한 것이다. 설계 단계는 기획, 요구사항분석, 기본설계의 프로세스가 있으며, 구현 단계는 기획, 상세설계, 구현, 테스트, 인수 프로세스가 있다. 단계별 프로세스의 주요 산출물은 Table 1과 같다.

Table 1. Software Step-by-step Ordering Process[10]

Step	Process	Deliverable Name
Design	Plan	Informatization Promotion Plan, Function Point Analysis, Bidding Preparation Documents
	Requirements Analysis	Requirements Specification, Use Case Scenario, User Validation/Verification
	Basic Design	Basic Design
Development	Plan	Informatization Promotion Plan, Detail RFP(Request For Proposal), Bidding Preparation Documents
	Detail Design	DB Design, Test Specification
	Development	Unit Module Design, Source Code, Test Scenario
	Testing	Test Result, Quality Review, Code Quality Report
	Deployment	Installation Result Sheet, Acquisition Plan, Pilot Operation Plan

소프트웨어 단계별 발주는 설계와 구현 단계가 있으며, 각 단계의 마지막은 사업종료 프로세스를 수행해야 한다. 정보화 사업 단계별 관리·점검 가이드[11]에 사업종료 프로세스는 사업 완료 검사, 인수 및 하자보수, 사업종료, 운영 및 유지보수 프로세스가 있다. 사업 완료 검사에서 발주사는 산출물 검사를 통지받은 날로부터 14일 이내에 완료해야 한다. 사업종료 프로세스에서 수행사가 관련 서류를 발주사에 제출하여 대가지급을 청구하면, 발주사는 청구받은 날로부터 근무일 기준 5일 이내로 대가를 지급해야 한다.

2.2 이더리움의 스마트 콘트랙트

비트코인은 암호화폐로 사용되고 있으나 이더리움은 스마트 콘트랙트를 사용하여 전자투표, 전자 계약서 등 다양한 기능을 제공하는 플랫폼 구현이 가능하다. 스마트 콘트랙트는 Solidity 프로그램을 사용하여 개발할 수 있으며, Solidity로 구축된 프로그램을 디앱(DApp)이라고 한다[12]. Fig. 2는 정수를 입력받고 반환하는 함수를 구현한 Solidity 예시이다.

스마트 콘트랙트는 사용자 간 합의된 계약서로 이더리움 블록의 트랜잭션에 함수와 데이터 집합으로 저장된다. 스마트 콘트랙트를 이용하여 계약자 간의 계약정보를 정의할 수 있으며 계약 조건이 만족하면 다음 프로세스를 진행하는 등의 신뢰성 있는 프로그램 개발이 가능하다[13].

3. 설계 및 구현

공공정보 소프트웨어 사업에서 대금을 자동으로 지급하기 위해서는 완료 조건을 정의해야 한다. 완료 조건은 발주사에

```
// SPDX-License-Identifier: MIT
pragma solidity ^0.8.10;  // Solidity Version

contract SmartContract{  // Contract Name
  uint data;  // Integer Parameter
  // Data Input
  function setData(uint _data) public{
      data = _data;
  }
  // Data Return
  function getData() public view returns(uint _data){
      _data = data;
  }
}
```

Fig. 2. Smart Contract Source Example

서 요청된 산출물이 전부 등록되고 승인되면 완료되었다고 볼 수 있다. 따라서 대금을 지급하고 산출물을 관리하기 위해서는 사업관리와 산출물 관리 서비스를 제공해야 한다.

이더리움 기반의 공공정보 소프트웨어 사업산출물을 관리하기 위해 이더리움 네트워크(Ethereum Network)와 산출물 관리 시스템(Deliverables Management System)을 설계한다. Fig. 3은 이더리움 기반 공공정보 소프트웨어 사업산출물 관리 시스템의 구조도를 나타내고 있다.

3.1 이더리움 네트워크

단일 서버/노드로 구성된 시스템은 장애 발생시 서비스를 제공하지 못하는 문제가 있다. 이더리움으로 다수의 노드를 구성하여 서비스를 제공하면 하나 이상의 노드에서 장애가 발생해도 연속적인 서비스 제공이 가능하다. 개발 시스템은 이더리움 재단이 공식적으로 제공하는 클라이언트 소프트웨어인 Geth(go-ethereum)를 사용하여 이더리움 네트워크를 설계한다. 이더리움 네트워크는 프라이빗(Private) 이더리움 구축과 추후 퍼블릭(Public)으로 확장을 위해 3개의 노드를 사용하게 설계한다. 이더리움 네트워크에 스마트 콘트랙트를 배포하여 블록체인에 산출물 정보를 저장하고 모든 산출물이 승인되면 대금 지급이 되도록 시스템을 구성한다.

1) 스마트 콘트랙트의 구성

스마트 콘트랙트는 프로젝트 정보와 산출물을 관리하고 베이스라인 정보를 저장할 수 있도록 설계한다. 이를 위해 5개의 스마트 콘트랙트로 구성하는데, 따로 구성하게 되면 스마트 콘트랙트 주소가 많아져 관리상 어려움이 발생한다. 따라서 관리의 편의성을 위해 메인 스마트 콘트랙트에서 연결된 스마트 콘트랙트를 호출하는 중앙집중식으로 Fig. 4와 같이 구성한다. 즉, ProjectContract는 프로젝트 정보를 관리하고 다른 스마트 콘트랙트를 통합 관리하며, SWProduct는 버전과 파일명 등의 산출물 정보를 관리한다. BusinessPartner는 계약정보와 수행사의 정보를 관리하는 스마트 콘트랙트이다. BaseLine과 BaseLineManagement 스마트 콘트랙트는 산출물의 최종본을 관리한다.

2) 이더리움 블록의 구성

발주사에서 산출물을 요청하고 수행사에서 산출물을 등록한 후 승인되는 과정을 Fig. 5로 표현하였다. 발주사가 블록체인에 스마트 콘트랙트를 배포(Deploy)하면 트랜잭션에 계약정보가 저장되고 산출물을 요청(sendTransaction)하면 요청된 정보가 저장된다. 이후 수행사가 요청받은 산출물을 등록하면 해당 정보 역시 블록체인의 트랙잭션에 저장된다. 이후에 산출물이 변경되면 수행사에서 산출물을 다시 등록한다. 이때 이전 산출물 해시값을 사용하여 신규 파일 해시값을 계산하며, 트랜잭션에는 이전 산출물 해시값과 신규 산출물 해시값이 같이 저장되어 산출물 이력 추적이 가능하다.

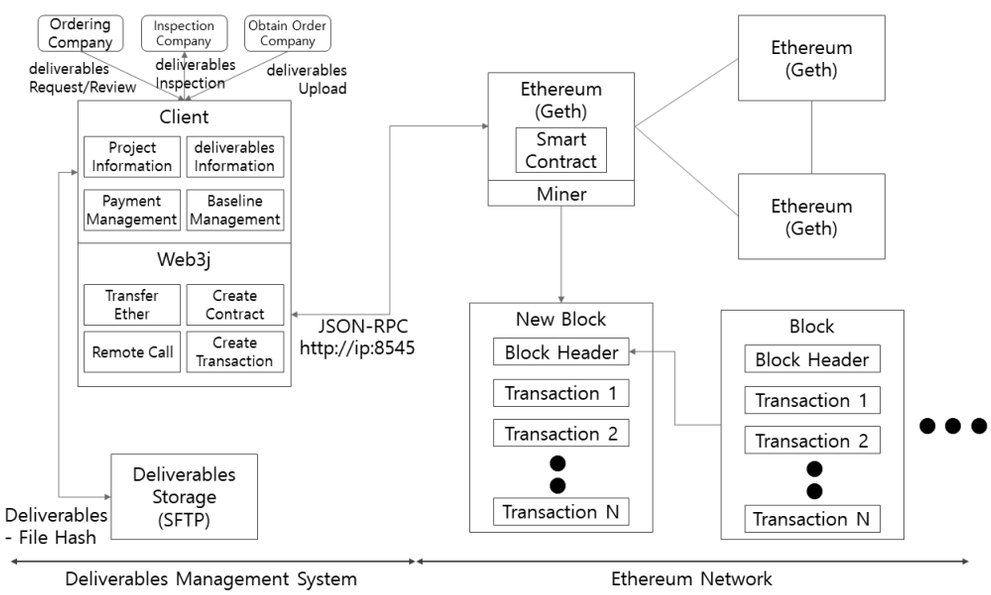

Fig. 3. System Structure

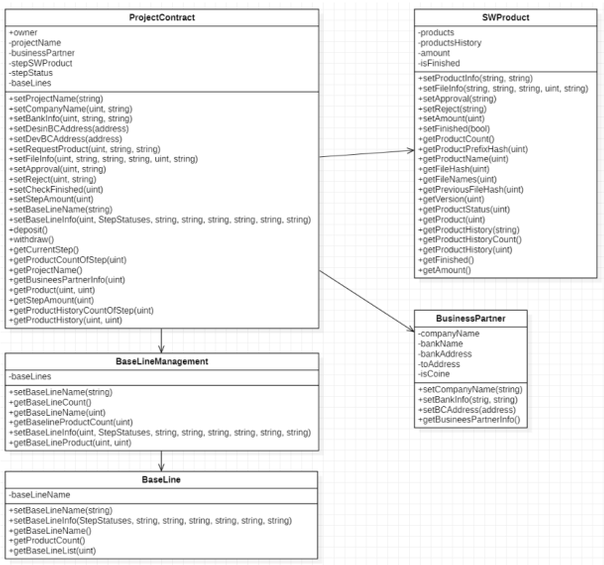

Fig. 4. Class Diagram of Smart Contract

3.2 산출물 관리 시스템

스마트 콘트랙트에서 제공하는 프로젝트 관리, 산출물의 요청/등록/승인, 베이스라인 관리 서비스를 제공하기 위해 클라이언트는 프로젝트 관리, 산출물 정보, 대금 관리, 베이스라인 관리로 구성한다. 이더리움 네트워크와 인터페이스를 위해 JSON-RPC를 제공하는 JAVA 라이브러리인 Web3j를 사용하고, 산출물을 저장하고 전송구간 보안을 위해 SFTP(Secure File Transfer Protocol) 서버를 사용하여 시스템을 설계한다.

Web3j의 인터페이스는 이더리움의 송금을 위한 Transfer Ether, 스마트 콘트랙트 생성을 위한 Create Contract, 스마트 콘트랙트에 저장된 정보를 읽어오는 Remote Call, 스마트 콘트랙트에 정보를 저장하는 Transaction으로 구성된다.

Algorithm : Finding Deliverable File Hash

Input. Step : Information Software Project Step
Type : Type of Deliverable
Version : Version of Deliverable
PreviousFileHash : Hash of Previous Deliverable File
File : Binary File of Deliverable
Output. FileHash : Hash of Deliverable File
Nonce : Big Integer
Using. FileHash: File Name of SFTP
Algorithm :
1 : **INITIALIZE** Prefix = step + type + version
INITIALIZE FileHash="", Nonce=0
2 : **WHILE** HEX(Prefix) != substring(FileHash,0,5) do
Nonce++
FileHash=HASH(File, PreviousFileHash, Nonce)

Fig. 6. Finding Deliverable File Hash Algorithm

3.3 산출물 해시 알고리즘 및 산출물 정보

소프트웨어 개발 사업에서 산출물은 변경이 자주 발생하기에 산출물 변경에 따른 산출물의 변경 추적성 관리 서비스가 필요하다. 따라서 본 논문에서는 버전 정보와 산출물 정보를 이용하여 계산된 산출물 해시값으로 산출물 변경 추적성을 관리하고자 한다. 구체적으로, 수행사에서 산출물을 등록할 때 산출물의 해시값을 계산하여 파일명으로 사용하고, 블록체인의 트랜잭션에 그 해시값을 저장한다. 트랜잭션에 저장된 해시값은 SFTP 서버에서 산출물을 가져올 때 사용한다.

산출물의 해시값 계산은 이전 산출물 파일의 해시값(Previous File Hash), 산출물 파일 자체(File), 논스(Nonce)를 이용하여 계산하며 해당 산출물에 대응되는 값으로 시작하는 해시값을 구할 때까지 반복한다. Fig. 6은 산출물 파일 해시값을 구하는 알고리즘으로 생성된 파일 해시값은 블록체인에 등록되며 SFTP 서버에 저장될 파일명으로 사용한다.

해당 산출물에 대응되는 값은 산출물 ID로부터 계산한다.

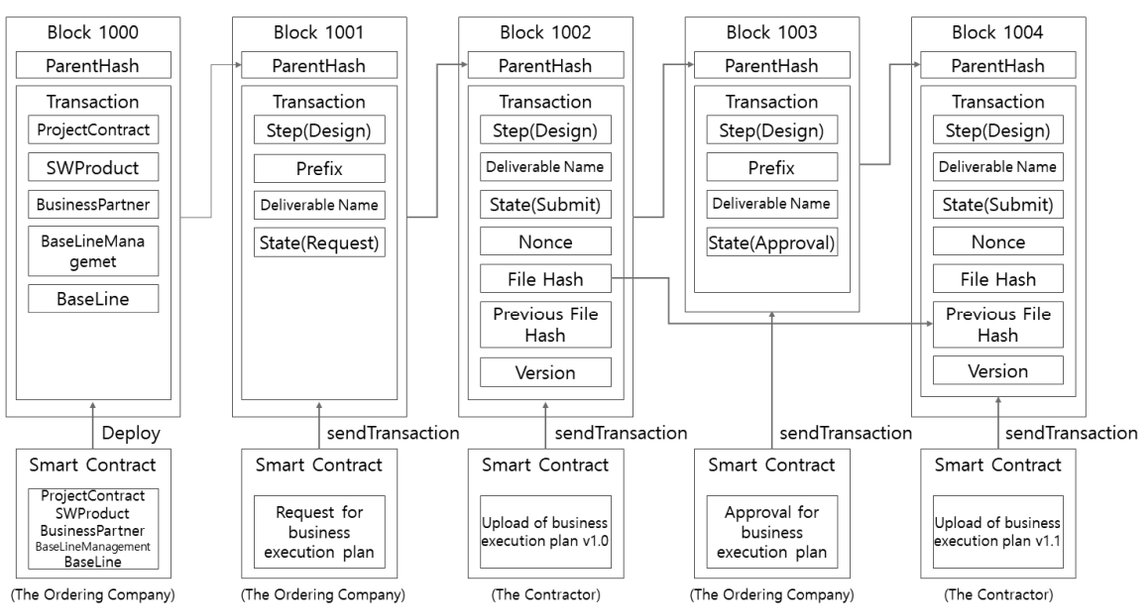

Fig. 5. System Block Configuration

Table 2. An Example of Software Design Step Deliverables

Process (step)	Deliverable Name (type)	ID	Hash
Plan (1)	Informatization Promotion Plan (01)	101011	18A93
	Function Point Analysis (02)	102011	18E7B
	Bidding Preparation Documents (03)	103011	19263
Requirements Analysis (2)	Project Implementation Plan (01)	201011	31133
	Requirements Specification (02)	202011	3151B
	Use Case Scenario (03)	203011	31903
	User Validation/ Verification (04)	204011	31CEB
Basic Design (3)	Class Specification (01)	301011	497D3
	Interface Specification (02)	302011	49BBB
	Architecture Design (3)	303011	49FA3

Table 3. An Example of Software Development Step Deliverables

Process (step)	Deliverable Name (type)	ID	Hash
Plan (4)	Informatization Promotion Plan (01)	401011	61E73
	Detail RFP (02)	402011	6225B
Detail Design (5)	User Interface Design (01)	501011	7A513
	DB Design (02)	502011	7A8FB
	Test Case (03)	503011	7ACE3
	Test Plan (04)	504011	7B0CB
Development (6)	Unit Module Design (01)	601011	92BB3
	Source Code (02)	602011	92F9B
	Test Scenario (03)	603011	93383
Testing (7)	Test Result (01)	701011	AB253
	Quality Review (02)	702011	AB63B
	Code Quality Report (03)	703011	ABA23
Deployment (8)	Installation Result Sheet (01)	801011	C38F3
	Acquisition Plan (02)	802011	C3CDB
	Pilot Operation Plan (03)	803011	C40C3

산출물 ID는 6자리인 10진수로, 단계별 프로세스(step) 1자리, 산출물 종류(type) 2자리, 버전(version) 3자리(Major 2자리, Minor 1자리)로 구성한다. 이때, 단계별 프로세스는 설계의 기획을 1, 요구사항 분석을 2, 기본설계를 3, 구현의 기획을 4, 상세설계를 5, 구현을 6, 테스트를 7, 인수를 8로 정의한다. 산출물 종류는 단계별 프로세스마다 다른데 요구사항 분석에서는 요구사항명세서를 01, Use Case 시나리오를 02 등으로 정의한다(Table 2와 Table 3 참고). 그리고 처음 작성된 산출물의 버전은 Major를 01, Minor를 0으로 하여 010으로 정의하고, 수정된 산출물의 경우 상황에 따라

Algorithm : Payment after Deliverable Approval

```
Input. ToAddress : Receive payment account
       Amount : Business performance
       ReqDelivers[] : Request Deliverable List
       AppDelivers[] : Approval Deliverable List
       IsPayed : Whether Payment is made
Algorithm :
1 : INITIALIZE IsFinish=true, IsCheck=false
2 : IF ReqDelivers.length not equal AppDelivers.length:
      IsFinish = false
3 : FOR ReqDelivers.length do
      IsCheck=false
      FOR AppDelivers.length do
          IF ReqDelivers.name equal AppDelivers.name:
              IsCheck=true
      IF isCheck equal false:
          IsFinish = false
4 : IF IsFinish equal true and IsPayed equal false:
      ToAddress.transfer(amount)
```

Fig. 7. Payment after Deliverable Approval Algorithm

Major 또는 Minor를 1씩 증가시키는 방법으로 정의한다.

6자리인 10진수 산출물 ID를 16진수로 변환하면 5자리(20 bits)로 변환된다. 변환된 16진수 5자리로 시작하는 해시값을 계산하여 스마트 콘트랙트에 산출물의 정보로 저장한다. Table 2와 Table 3은 버전이 1.1인 경우 프로세스별 산출물 해시값의 예시이다.

3.4 이더리움 송금

발주사가 요청한 산출물이 모두 등록되고 승인되면 자동으로 대금을 지급한다. 발주사가 산출물을 승인할 때마다 Fig. 7의 자동 대금 지급 알고리즘을 통해 대금 지급 여부와 산출물이 전부 승인되었는지 확인을 한다. 모든 산출물이 승인되고 대금이 지급되지 않았으면 수행사로 대금을 자동으로 송금한다. 스마트 콘트랙트에서 송금은 발주사가 스마트 콘트랙트로 대금을 먼저 입금하고, 입금된 대금을 스마트 콘트랙트가 수행사로 송금한다.

대금 지급은 ProjectContract 스마트 콘트랙트의 set CheckFinished 함수에서 수행한다. 해당 프로세스의 모든 산출물이 승인되었는지 확인하고 모두 승인되었다면 수행사의 이더리움 주소 정보를 조회하고 발주사가 스마트 콘트랙트에 송금한 대금을 발주사로 송금하게 된다.

4. 검증 및 분석

본 장에서는 이더리움 기반 공공정보 소프트웨어 사업산출물 관리 시스템을 검증하고 분석한다. 검증을 위한 시스템 구현은 프라이빗 이더리움 기반에 솔리디티와 자바로 개발하였고, 하드웨어는 i5-10210U CPU @ 1.60GHz, 운영체제는 Windows 10 환경에 설치하여 실행하였다. 개발에 사용한 프로그램의 목록은 Table 4와 같다.

Table 4. Software List

Software	Version	Description
Geth	1.10.6	- Private Ethereum
web3j	4.3.0	- Communication between Geth and Client
solidity	0.7.0	- Smart Contract Development Tool
Remix		- Solidity WEB IDE
JAVA	1.8.0	- Client Development Language
Eclipse	2021-03 (4.19.0)	- Client Development IDE
SFTP	freeFTPd 1.0.13	- SFTP Server

Geth는 이더리움을 구축할 수 있는 오픈소스 프로젝트이며, web3j는 이더리움 노드와 클라이언트 프로그램 간의 인터페이스 역할을 담당한다. Solidity는 스마트 콘트랙트를 개발할 수 있는 언어이며, Remix는 Solidity IDE 도구이다. Eclipse 기반에서 JAVA Swing으로 개발을 진행하였으며 SFTP 서버에 산출물을 저장한다.

4.1 블록체인 네트워크 구축 및 실행

프라이빗 블록체인 노드를 구축하기 위해 Fig. 8과 같이 제네시스 블록(genesis.json) 파일을 생성한다. config 헤더는 제네시스 블록의 설정값을 정의하는 것으로, chainId는 블록체인을 식별하는 정숫값이고 homesteadBlock는 하드포크 블록이다. EIP(Ethereum Improvement Proposal)는 이더리움 개선 제안으로 숫자를 사용하여 구분한다. eip150Block는 최대 가스(Gas) 비용을 변경한 설정이며, eip155Block는 replay attack을 막기 위한 설정이다. eip158Block는 State Cleaning

```
{
  "config": {
    "chainId": 15,
    "homesteadBlock": 0,
    "eip150Block": 0,
    "eip155Block": 0,
    "eip158Block": 0
  },
  "nonce": "0x0000000000000042",
  "mixhash": "0x0000000000000000000000000000000000000000
              0000000000000000000000000000000000",
  "difficulty": "0x00",
  "alloc": { },
  "coinbase": "0x0000000000000000000000000000000000000000
              000000",
  "timestamp": "0x00",
  "parentHash": "0x0000000000000000000000000000000000000000
                0000000000000000000000000000000000",
  "extraData": "0x00",
  "gasLimit": "0xe8d4a50fff"
  }
}
```

Fig. 8. Genesis Blocks (genesis.json)

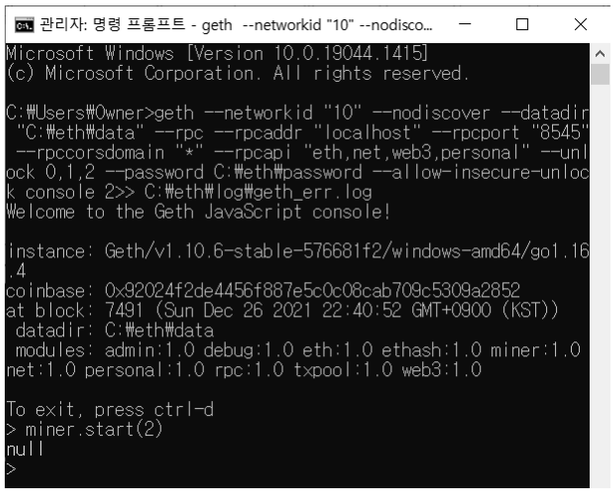

Fig. 9. Ethereum Master Node Execution

관련 설정이다. nonce와 mixhash는 블록에 충분한 양의 작업 증명 연산을 실행하였음을 의미하며, difficulty는 블록의 생성 난이도를 의미한다. gasLimit는 사용 가능한 가스의 최대크기이다[14-16].

이더리움의 시작 블록을 Fig. 8과 같이 설정하고 Fig. 9와 같이 실행한다.

4.2 시나리오 기반 검증

본 논문에서 제안한 시스템을 검증하기 위해 클라이언트를 통해 Table 5의 시나리오로 진행한다. 수행사에서 등록한 파일의 해시값이 설계의 산출물 해시값의 prefix 값이 같은지 확인하고, SFTP 서버에 저장된 파일명이 이더리움 블록체인에 등록된 해시값과 같은 값인지 확인한다. 그리고 모든 산출물이 승인된 후 수행사로 대금이 송금되었는지 확인한다.

1) 수행사 정보를 입력

수행사의 정보와 지갑 주소(0x22642145436e2ee0c32b26f756f12b2eedd1d410)를 Fig. 10과 같이 입력하여 프

Table 5. System Scenario

Scenario Sequence	Description
1	- Enter the performer's information.
2	- Enter the amount of payment in stages.
3	- Enter the requested deliverables at the ordering company.
4	- The performer upload the deliverables.
5	- All deliverables are approved by the ordering company.
6	- Check whether the payment has been made from the ordering company to the performer.
7	- Check if the baseline is set.
8	- Check the file list in SFTP server.

Fig. 10. The Contractor Information

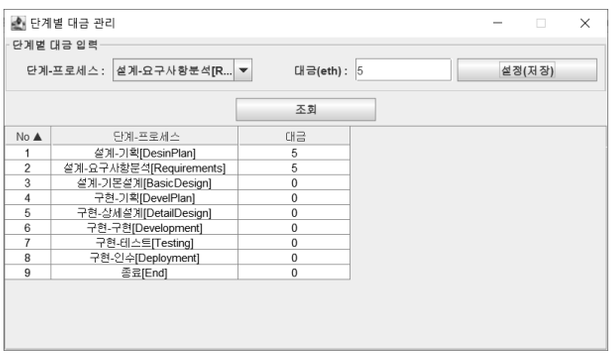

Fig. 11. Step-by-Step Payment

Fig. 12. Request Deliverables

로젝트 정보를 관리한다. 지갑 주소는 발주사에서 모든 산출물을 승인하였을 때 수행사에서 대금을 받는 이더리움 주소이다.

2) 단계별 지급하는 대금을 입력

프로젝트 정보로 Fig. 11과 같이 설계 단계의 기획과 기본설계 프로세스에 각각 5 ether를 대금으로 입력하였다. 발주사에서 모든 산출물이 승인되었을 경우 5 ether가 정상 지급되어야 한다.

3) 발주사에서 요청산출물을 입력

Fig. 12는 발주사에서 수행사에 산출물을 요청하는 화면으로, 요청하는 산출물을 단계별로 입력한다. 산출물 코드는 3자리로 단계별 프로세스 1자리와 산출물 2자리로 구성된다. 설계 단계의 요구사항분석 프로세스(2)에서는 사업수행계획서(01)와 요구사항명세서(02)를 요청하였고, 각 산출물 코드는 201(사업수행계획서)과 202(요구사항명세서)가 된다.

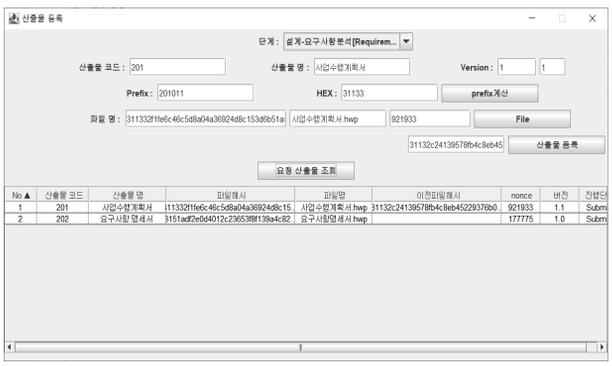

Fig. 13. Upload Deliverables

Table 6. Results of the Finding Deliverable File Hash Algorithm

Version	1.0	1.1	1.2	1.3	1.4
nonce	852871	1413581	553455	73059	1323801
run time (ms)	1678	2795	1130	199	2525

4) 수행사에서 산출물을 등록

Fig. 13은 수행사가 산출물을 등록하는 화면이다. 사업수행계획서의 경우 산출물의 코드가 201이고, 버전은 1.1이므로 산출물 ID는 201011이며 이의 Hex값은 31133이다. 이전 파일의 해시값은 31132c24139578fb4c 8eb45229376b0d9d350 90a3ebef906e04812313a5a3801이며 계산된 논스는 921933이다. 등록된 파일의 해시는 311332f1fe6c46c5d8a04a3 6924d8c153d6b51a599999602e07410a5f81e0789로 31133으로 시작됨을 확인하였다.

수행사에서 산출물을 업로드하면 파일 해시값과 논스값은 Fig. 6의 알고리즘에 따라서 계산된다. Table 6은 파일의 크기가 1.01MB인 사업수행계획서를 사용한 결과로, 동일한 파일을 이용하여 1.0 버전부터 순차적으로 버전을 올렸을 때 수백 밀리초부터 수 초까지의 시간이 소요되었다. 수행시간이 클수록 계산된 논스값이 크다는 것을 알 수 있다.

5) 발주사에서 모든 산출물을 승인

Fig. 14는 수행사가 등록한 산출물을 발주사에서 승인하는 화면이다. 발주사에서 등록된 산출물을 모두 승인하면 Fig. 11에 등록한 대금을 수행사에 이더리움으로 지급하게 된다.

6) 발주사에서 수행사로 대금이 지급되었는지 확인

Fig. 15는 수행사의 지갑 주소로 이더리움 계좌 잔액을 조회하였다. 첫 번째 조회 결과는 프로젝트 시작 전에 약 18 ether가 있음을 확인한 것이며, 두 번째 조회 결과는 설계-기획 단계가 종료된 후 5 ether가 송금된 약 23 ether가 잔액으로 있음을 확인한 것이다. 마지막 조회 결과로 설계-요구사항분석 단계에서 5 ether가 송금된 약 28 ether가 잔액으로 있음을 확인하였다.

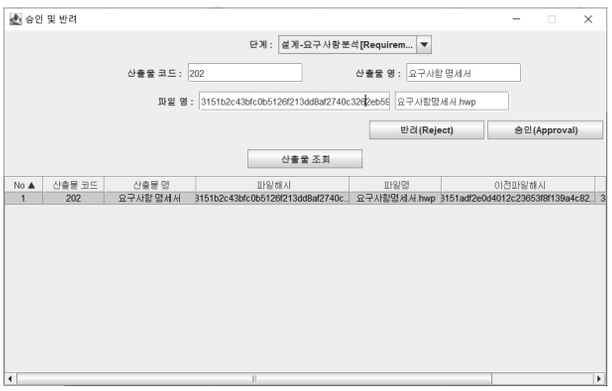

Fig. 14. Approval of Deliverables

Fig. 15. Payment Confirmation

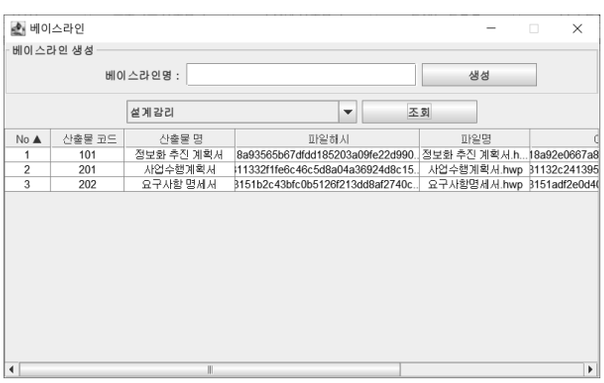

Fig. 16. Baseline

7) 베이스라인 설정 확인

Fig. 16에서 감리사에 제공하는 산출물 목록을 베이스라인으로 관리 할 수 있다. 베이스라인을 생성하면 현재 승인 완료된 산출물 목록을 자동으로 생성한다. 발주사에서 요청 산출물은 “정보화 추진계획서”, “사업수행계획서”, “요구사항 명세서”이며, 모두 승인되어 베이스라인으로 생성되었다.

8) SFTP 서버 파일 목록 확인

Fig. 17은 SFTP 서버 접속화면으로 수행사에서 등록된 산출물 파일 리스트를 확인하는 화면이다. 산출물 파일의 20 bits(5자리) 문자열을 확인하면 산출물과 버전 정보 확인이 가능하다. 위에서 4번째 파일은 31133으로 시작하며, 10진수로 변환하면 201011이 된다. 즉, 2번째 설계-요구사항분석 프로세스에서 01번째 사업수행계획서의 1.1버전임을 알 수 있다. 수행사에서 등록한 사업수행계획서 1.1버전의 해시값 11332f1fe6c46c5d8a04a36924d8c153d6b51a599999602e07410a5f81e0789와 같음을 확인하였다.

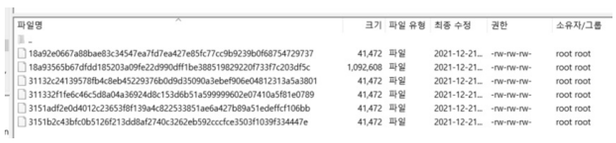

Fig. 17. SFTP Server File List

Table 7. Comparison of Deliverables Management Methods

	Web-base Project Management System[5]	ChainSoft [7]	Proposal System
Share Deliverables	○	○	○
Deliverables Traceability	X	○	○
Payment	X	△	○
Source Configuration Management	X	○	X
Distributed System	X	○	○
Step-by-step Deliverables Management	X	X	○
Deliverables Apploval	X	X	○
Baseline	X	△	○

4.3 분석

본 논문에서 제시하는 공공정보 소프트웨어 사업산출물 관리 시스템은 이더리움 블록체인으로 구현하여 기존의 중앙집중화된 데이터베이스를 사용하는데 생기는 단점인 SPOF (Single Point Of Failure), 운영의 불투명성 같은 문제를 해결할 수 있다. 분산 네트워크로 구성되어 이더리움 노드 중 장애가 발생하여도 나머지 노드가 정상 동작하므로 시스템 장애에 강점이 있다. 노드에서 모든 블록은 이전 블록의 해시값을 참조하며, 산출물의 추적성을 관리하기 위해 이전 산출물의 해시값을 다음 버전의 산출물 해시값에 사용하여 데이터 변조의 위험에 대응할 수 있다.

이더리움 기반 공공정보 소프트웨어 사업산출물 관리 시스템(이하, 제안 시스템)의 평가는 웹 기반 프로젝트 관리 시스템[5]과 ChainSoft[7]를 대상으로 8가지 항목에 대해 Table 7과 같이 비교 분석하였다.

(1) 산출물 공유(Share Deliverables): 산출물은 적합한 권한을 가진 발주사/수행사/감리사 사이에 공유되어야 한다. 세 시스템 모두 산출물 공유 기능을 제공하며, 특히 제안 시스템은 블록체인을 통해 산출물의 등록 정보를 투명하게 관리할 수 있다.

(2) 산출물 추적성(Deliverables Traceability): 산출물은 이전 버전의 산출물을 찾을 수 있어야 한다. 제안 시스템은 이전 산출물 해시값을 사용하여 산출물 해시값을

계산하고 블록에 이전 산출물의 해시값을 같이 저장하여 이전 버전의 산출물 추적이 가능하다. ChainSoft는 GitHub를 사용하여 산출물 추적성을 제공한다.

(3) 대금 지급(Payment): 모든 산출물이 승인되면 대금을 지급해야 하고, 대금 지급을 자동화하여 발주사와 수행사 간 갈등을 줄일 수 있다. 제안 시스템은 발주사에서 산출물을 승인하면 스마트 콘트랙트를 통해 모두 승인되었는지 확인하고 수행사에 대금을 자동으로 지급한다. ChainSoft는 모든 산출물이 아닌 소프트웨어에 대한 검증만 이뤄지면 대금을 지급한다.

(4) 소스 형상 관리(Source Configuration Management): 프로그램 소스의 변경 이력을 관리한다. GitHub를 사용한 ChainSoft는 소스 형상관리가 가능하다. 제안 시스템은 제출된 산출물을 승인하고 변경 이력은 해시값으로 관리하는 시스템으로 소스의 형상관리는 지원하지 않으나 프로그램 소스를 압축파일로 관리 할 수 있다.

(5) 분산 시스템(Distributed System): 분산 시스템을 지원하여 장애에 대응해야 한다. 제안 시스템은 3개의 이더리움 노드를 사용하여 장애 발생 시 정상 서비스를 제공할 수 있다. ChainSoft는 GitHub와 블록체인을 사용하여 분산 서비스가 가능하다.

(6) 단계별 산출물 관리(Step-by-step Deliverables Management): 소프트웨어 개발 단계별로 제출 산출물을 관리해야 한다. 제안 시스템은 개발 단계별로 발주사가 산출물을 요청하고 수행사는 등록할 수 있어 단계별 관리가 가능하다.

(7) 산출물 승인(Deliverables Apploval): 등록 산출물을 승인해야 한다. 제안 시스템은 수행사가 등록한 산출물에 대해 발주사에서 승인할 수 있다.

(8) 베이스라인(Baseline): 프로젝트 상황에 따라 단계 완료 시점이나 산출물 제출 시점을 정할 수 있어야 한다. 제안 시스템은 산출물을 제출이나 단계별로 베이스라인 설정이 가능하다. ChainSoft는 베이스라인 기능을 직접적으로 제공하지 않지만, 베이스라인에 포함되는 산출물만 따로 등록하면 유사하게 관리할 수는 있다.

5. 결 론

본 논문에서는 이더리움 기반 공공정보 소프트웨어 사업산출물 관리 시스템을 제안하였다. 이 시스템은 사업산출물 정보를 스마트 콘트랙트를 이용하여 이더리움 블록체인에 저장한다. 발주사는 요청된 산출물이 수행사에서 등록하였는지 확인 가능하며, 산출물이 모두 승인되면 계약된 대금이 수행사의 이더리움 주소로 자동 송금된다.

수행사는 요청된 산출물의 검수가 완료되어야만 대금이 지급되므로 산출물을 등록해야 하고, 발주사는 수행사가 제출한 산출물을 등록 순서대로 승인할 수 있어 점검 시간 확보가 가능하다. 이더리움의 분산 환경에서 서비스를 제공하여 하나 이상의 노드에 장애가 발생해도 정상 서비스가 가능하다. 산출물 누락 여부 확인이 쉬우며, 모두 승인되면 자동으로 대금 지급이 이루어져 내부 프로세스가 간단해진다. 산출물의 정보가 블록체인에 등록되므로 산출물의 이력 관리가 투명해지고 최종 버전의 산출물 확인 또한 쉬워진다. 산출물 ID에 기반한 해시값으로 산출물을 관리하여 식별 및 분류를 쉽게 할 수 있으며, 산출물의 버전에 따라 해시값이 순차적으로 증가하므로 산출물의 버전관리가 쉬워진다.

향후 연구로 형상관리 시스템과 연동으로 소스 형상을 관리하고, 퍼블릭 이더리움으로 확장하여 환전 가능한 이더를 송금하는 환경을 구축하고자 한다. 퍼블릭 이더리움으로 전환하기 위해서는 많은 테스트가 필요하며 SFTP 서버에 승인된 사용자만 접속할 수 있도록 보안 이슈도 해결해야 한다.

References

[1] S. Nakamoto, “Bitcoin: A peer-to-peer electronic cash system,” 2008.

[2] Y. K. Yang and S. H. Chun, “A study on the utilization status and development plan for blockchain technology: Focusing on cryptocurrency policies of foreign countries,” *The Journal of Business Education*, Vol.33, No.2, pp.47-70, 2019.

[3] E. J. Kim and H. C. Kim, “A study on IT project deliverables of owners’ viewpoint -Focuses on small-scale projects in public institutions,” in *Proceedings of the Korea Information Processing Society Conference*, pp.605-608, 2014.

[4] Ministry of the Interior and Safety·NIA, “e-Government Support Project ISP Deliverables Inspection Guide,” 2015.

[5] J. Y. Kim, H. S. Kim, C. Y. Jin, Y. M. Hwang, S. R. Kim and B. M. Kim, “Implementation of web-based project management system,” in *Proceedings of KIIT Conference*, pp.556-559, 2021.

[6] S. M. Lee and S. U. Lee, “Configuration management technique for unity-based game development projects,” in *Proceedings of Korea Software Congress 2020*, pp.1224-1225, 2020.

[7] M. Król, S. Reñé, O. Ascigil, and I. Psaras, “ChainSoft: Collaborative software development using smart contracts,” in *Proceedings of the 1st Workshop on Cryptocurrencies Blockchains Distributed Systems (CryBlock)*, pp.1-6, 2018.

[8] N. Nizamuddin, K. Salah, M. A. Azad, J. Arshad, and M. H. Rehman, “Decentralized document version control using ethereum blockchain and IPFS,” *Computers and Electrical Engineering*, Vol.76, pp.187-197, Jun. 2019.

[9] Ministry of Science and Technology Information and Communication, “Software Promotion Act 17348,” 2020.

[10] NIPA, “Software step-by-step ordering guide,” 2019.

[11] Ministry of Science and ICT·Ministry of the Interior and Safety·NIA, “Step-by-step management and inspection guide for informatization business V3.0,” 2015.
[12] Ethereum [Internet], https://ethereum.org.
[13] Solidity Programming Language [Internet], https://soliditylang.org.
[14] Explaining the Genesis Block in Ethereum [Internet], https://arvanaghi.com/blog/explaining-the-genesis-block-in-ethereum.
[15] Ethereum private network configuration guide [Internet], https://gist.github.com/0mkara/b953cc2585b18ee098cd#file-genesis-md.
[16] EIPs/EIPS at Master [Internet], https://github.com/ethereum/EIPs/tree/master/EIPS.

이 은 주
https://orcid.org/0000-0002-6070-8049
e-mail : brent@knou.ac.kr
2004년 인제대학교 정보통신공학과(학사)
2020년 ~ 현 재 한국방송통신대학교 정보과학과 석사과정
2018년 ~ 현 재 한국전력기술 스마트융합실 선임
관심분야 : 블록체인, 빅데이터, 정보보호

김 진 욱
https://orcid.org/0000-0003-4986-0848
e-mail : gnugi@knou.ac.kr
1998년 서울대학교 수학과(학사)
2000년 서울대학교 컴퓨터공학과(석사)
2006년 서울대학교 전기・컴퓨터공학부 (박사)
2013년 ~ 현 재 한국방송통신대학교 컴퓨터과학과 교수
관심분야 : 컴퓨터이론, 알고리즘, 생물정보학, 정보보호