International Journal of Engineering Trends and Technology
ISSN: 2231–5381 / https://doi.org/10.14445/22315381/IJETT-V73I2P110

Volume 73 Issue 2, 119-134, February 2025
© 2025 Seventh Sense Research Group®

Original Article

Formal Development of Blockchain Enabled E Healthcare System Using Event-B

Raghuraj Singh Suryavanshi[1], Vishal Nagar[2], Abhishek Rawat[3], Divakar Yadav[4]

[1,2,3]Department of CSE, Pranveer Singh Institute of Technology, Kanpur, India.
[4]Department of CSE, Institute of Engineering and Technology, Lucknow, India.

[3]Corresponding Author : abhi.rawatcse@gmail.com

Received: 05 September 2024 Revised: 10 January 2025 Accepted: 27 January 2025 Published: 21 February 2025

Abstract *- Developing complicated systems makes correct software development a difficult process. Software may contain imperfections as a result of incorrect specifications. Traditional testing procedures are insufficient to ensure the proper functioning of a complicated system. Formal approaches are essential to capture accurate system requirements and accurate logic about the challenges. Formal methods are a mathematical approach that provides issued specifications, solutions, and verification of correctness. This paper presents the formal verification of the Blockchain Enabled Electronic Health Record System using Event-B, implemented in Remix IDE using Blockchain Technology. Blockchain technology ensures transparency and integrity while securely storing and exchanging patient data. Its decentralized structure can facilitate smooth interoperability between healthcare providers, improve data security, and streamline operations. Event B is a formal technique for system-level modeling. It is a mathematical and formal notation that allows complicated systems to be specified, refined, and verified. This paper aims to use blockchain technology and a formal verification process for developing the Electronic health record system with proof of obligation and provide secure data storage by defining easy access rules for the users of the proposed framework. This work is to provide the electronic health record system with the benefits of scalable, secure, transparent, and integral blockchain-based solutions.*

Keywords *- Blockchain technology, E-Healthcare system, Event-B, Formal methods, Smart contract.*

1. Introduction

Data about the healthcare system is very important to everyone. It keeps several records, such as general patient data, clinical decision support data, order entry, and health information exchange data. The recent advances in technology are affecting all parts of human life that are changing the way of living. Communication technology and blockchain are the main enablers for the decentralization of the healthcare system, offering a new, modern, digital ecosystem to patients and healthcare providers [1]. Electronic Health Record (EHR) systems must transform the healthcare system from the most centralized industry to a decentralized industry, and it will also help doctors provide high-quality care.

Inter-institutional EHRs are increasingly being used to improve the effectiveness of health care and offer accurate information [2]. It is easy to argue that the three security goals of confidentiality, data transparency, integrity, and data availability are essential components of the EHR system. For distributed database systems, there is a requirement to explicitly validate and confirm the correctness of the framework. Traditional testing procedures are ineffective in ensuring the accuracy of such systems [3]. In recent years, formal verification technology has advanced significantly with the rapid advancement of computer science, particularly the appearance of participatory theorem-proving tools such as Coq, Isabelle/HOL, and others [4]. Formal methods are mathematical procedures that specify and reason about system attributes by employing notions and ideas from mathematics and formal logic. It provides a framework for writing specifications, analyzing, and verifying models methodically. The formal approaches allow for examining the system requirements, design, and system behavior, including the possibility of failures. The technologies also offer automated proofs for system property verification [5].

Furthermore, in this paper, the Event-B technique is employed. The event-B method is a formal approach to system-level modeling and analysis. With applications in critical systems such as the Ariane 5 rocket and tool support through platforms like the Rodin Tool [6], Event-B offers a rigorous and formalized framework for ensuring the reliability and correctness of complex systems. The context and the machine are the two primary components of an Event-B specification in the larger picture of formal methods and the Event-B methodology [7].

- Context: The context in Event-B reflects the system being modeled's static component, providing the sets, constants, and axioms that explain the system's beginning circumstances and limitations.
- Machine: The machine in Event-B illustrates the system's dynamic behavior, specifying the elements, instances, and invariants that describe the system's state transitions and operations.

To provide procedures and tools to improve the accuracy of contracts for EHR systems by formally validating the Solidity contract's capabilities and security requirements by converting it to the relevant Event-B model. The structure of this paper is as follows: Section 2 briefly outlines the Literature Review, Section 3 depicts the Problem Statement, Section 4 illustrates the blockchain and its Smart Contract, Section 5 presents the Event-B Method, Section 6 briefly describes the Relation Between Blockchain and Event B in EHR System, Section 7 presents the Different Use Cases of EHR Systems, Section 8 depicts the Data flow of Blockchain-Enabled E Healthcare Systems and Sect. 9 present verification of solidity contracts in Event-B. Section 10 defines formal verification of the solidity contract of blockchain-based E-Healthcare system using Event-B, Section 11 describes the result and discussion of this research, and finally, Section 12 concludes the paper.

2. Literature Review

Numerous scholarly articles have explored the use of blockchain for EHR Systems. These papers highlight the advantages and challenges associated with implementing blockchain. Here are some notable highlights from the related work. Ekblaw A. et al. (2016) [8] explored the blockchain to identify the challenges of privacy, security, and interoperability in EHRs. The paper introduces "MedRec," a decentralized EHR system built on blockchain. Alzahrani, F. A . et al. (2021) [9] provided an in-depth analysis of blockchain's applications in healthcare. The paper discusses the potential of blockchain to revolutionize EHR systems, highlighting its ability to enhance interoperability, data security, and patient privacy. Kubendiran M. et al. (2019) [10] presented a study on using blockchain to facilitate the secure sharing of EHR. The paper discusses the implementation of healthcare data gateways using blockchain technology, emphasizing its role in ensuring data integrity, privacy risk control, and secure access to EHR across different healthcare providers. One of the author explored the security aspects of blockchain applications of IoT, with a focus on decentralized EHR systems. The paper addresses the opportunities of integrating blockchain into EHR systems, emphasizing the potential for enhanced privacy, data security, and data delivery among healthcare professionals. Chandini A. et al. [11] focused specifically on verifying smart contracts using pre-transaction signatures in blockchain-based EHR systems. In this domain, it investigates the use of pre-transaction signature verification and validates the accuracy and security features of smart contracts. Almakhour M et al. [12] highlighted a summary of the techniques for verifying smart contracts. Before being used, a smart contract should function as intended because the data stored on the blockchain is unchangeable. Tolmach P. et al. [13] illustrated formal methods and verification techniques for contracts.

Additionally, characteristics across several domains were compiled and connected to the capabilities of currently available verification tools. Wang Y et al. [14] describe VERISOL as a verification tool for detecting bugs in smart contracts. However, it can only be applied to a particular smart contract that carries out the underlying workflow policy represented by finite state machines; it cannot be applied to a general smart contract. Furthermore, the model testing approach will encounter a state explosion when a given system becomes extremely complex. The described technique for validating general Solidity contract qualities is based on first-order theorem proving.

Garfatta I. et al. [15] describe the soundness of Solidity contracts was verified through the application of a theorem-proving technique to translate Formal verification. This is a fascinating and innovative area of research, but it is highly challenging due to factors like dependent types, interaction with SMT provers, and monads for side effects. An even simpler proof technique is available for Dafny, another imperative language. Leonardo Alt et al. [16] described the contract code's compliance with the requirements outlined by the "require/assert" statements, which has been established through the use of the SMT solver, a formal verification tool that automatically verifies the module. It attempts to demonstrate the conditions included in "assert" statements are always true by treating them as assumptions. The SMT solver provides the user with an alternate example illustrating the possible violation scenario if a claim is rejected.

3. Problem Statement

After studying multiple publications, a range of strategies exist for establishing the EHR system. Furthermore, a detailed review of blockchain literature demonstrates that various researchers have successfully implemented the EHR System. Introducing blockchain technology improves transparency and robustness due to the network's decentralized structure. Existing research, however, has several serious problems, including computational complexity, longer processing times, data leaks, vulnerability to different types of attacks, and scalability issues. Additionally, because blockchain is unchangeable, any errors or bugs will be irreversible once they are published, which could result in significant financial losses. Formal verification methods have been suggested to improve blockchain's reliability and security. These approaches can be used to evaluate distributed databases, and because blockchain is distributed, they have the possibility for formal verification.

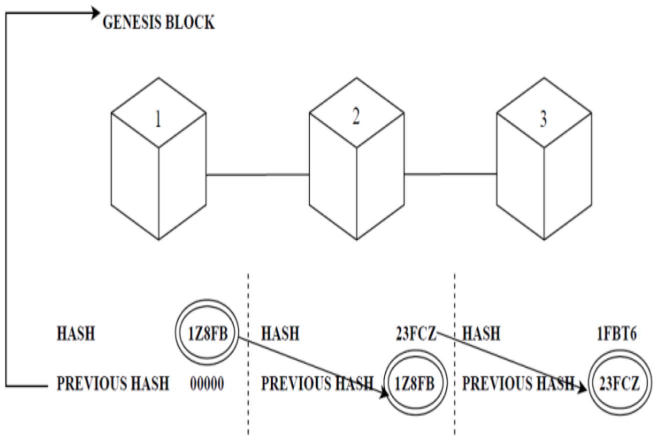


Fig. 1 Structure of the blocks

4. Blockchain and Smart Contract

Blockchain technology, most famously linked to digital currencies like Bitcoin, has developed into a broad field with many uses. Blockchain is a safe and unchangeable record of every transaction produced by each block in the chain, which has a list of actions and is connected to the block before it (Figure 1) [18]. Because of its security, transparency, and resistance to change, this technology has been used in deciding systems and medical records, among other applications.

Smart contracts are contracts that execute on their own and include the conditions of the buyer-seller contract directly encoded into code. Because of this automation, these terms may be automatically enforced and carried out, thus lowering the need for middlemen and increasing the security and effectiveness of transactions. Smart contracts can automate and digitize processes without requiring human intervention because they are built to run on a blockchain [19]. They are essentially tamper-proof programs that execute actions when specific conditions are met, such as automatically transferring ownership of a token or asset when payment is received. This capability not only enhances trust among the parties involved but also ensures that the contract remains intact even if one party detaches from the network [20]. With smart contracts, a diverse range of transactions and agreements can be executed, from simple financial transfers to complex multiparty arrangements, all while guaranteeing security, trust, and accuracy.

4.1. Advantages of Blockchain in EHR System

Blockchain technology offers several significant advantages for EHRs, enhancing their efficiency, security, and interoperability.

- Enhanced Security and Privacy
- Improved Interoperability
- Increased Transparency and Trust
- Patient Empowerment and Control
- Cost Reduction and Efficiency

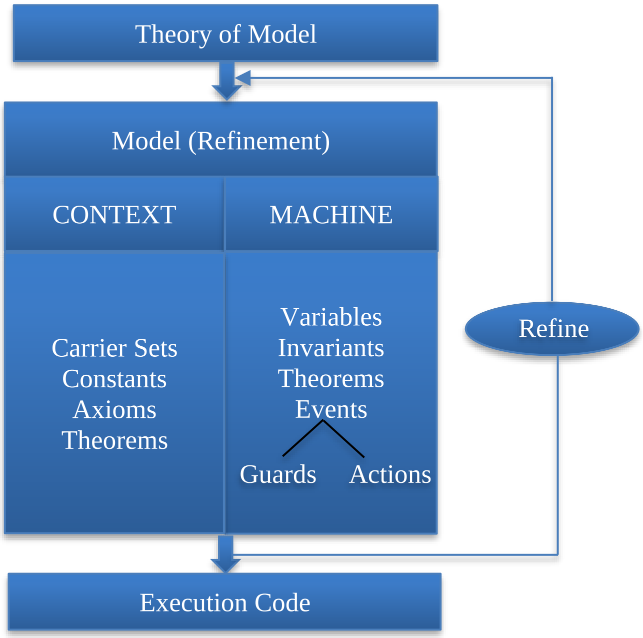


Fig. 2 Event-B components

5. Event-B Method

The Event-B approach is a popular formal modeling and verification method in the area of software engineering. This method is well-known for its methodical and rigorous approach to building and analyzing complex systems. At its core, Event-B uses a gradual refinement technique to gradually add system details into the formal model [21].

This method ensures that the system is built in a controlled and systematic manner and allows the design to progress from abstract principles to actual implementations. Event B enables the modular development of complex systems by refining the model [22]. One of the important characteristics of Event-B is the application of proof obligations to assure the model's consistency and accuracy. These proof responsibilities describe properties and limitations that the system must meet, and they play an important part in the verification procedure. The context and the machine are the two primary components of an Event-B specification in the larger picture of formal methods and the Event-B methodology, as shown in Figure 2.

5.1. Advantages of Event B Method in Blockchain Enabled EHR System

The Event B method for software development can significantly enhance blockchain-enabled EHRs. Here are the key advantages:

- Formal Specification
- Improved Verification and Validation
- Enhanced Modularity and Reusability
- Support for Incremental Development
- Increased Stakeholder Involvement

6. Relation Between Blockchain & Event B in EHR System

The relationship between blockchain technology and Event B in EHR systems is pivotal for enhancing data integrity, security, and interoperability. Blockchain provides a distributed and unchangeable ledger that ensures data remains tamper-proof, while Event B offers a formal method for specifying and designing system behaviors, particularly in how patient records are created, modified, and accessed [23]. By integrating these two technologies, EHR systems can log every event that changes patient data, creating a complete audit trail and allowing secure decentralized access without a central authority. This combination enhances security through encryption and consensus mechanisms, models the necessary protocols for controlled access, and facilitates interoperability between different EHR systems.

7. Different Use Cases of EHR Systems

Electronic Health Record (EHR) systems serve a multitude of purposes in the healthcare landscape, enhancing patient care and operational efficiency across various settings [24].

7.1. Patient Registries

EHRs are increasingly utilized to support patient registries, which collect and analyze patient-level clinical information for specific diseases or conditions.

7.2. Care Coordination

In multi-speciality clinics, EHR systems facilitate care coordination by ensuring that relevant health information is shared seamlessly among various stakeholders, including doctors, nurses, caregivers, and patients.

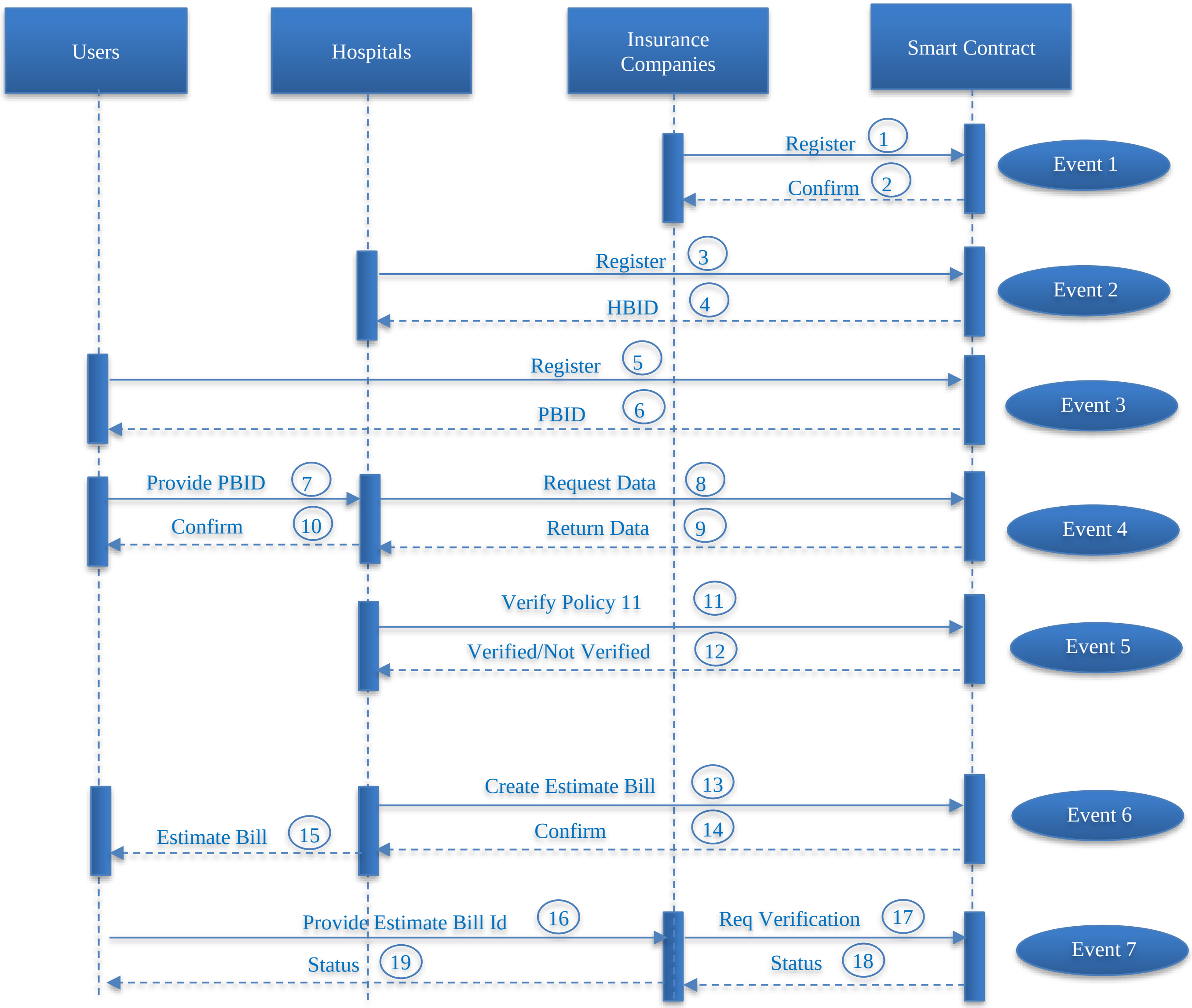


Fig. 3 Sequence diagram of blockchain enabled E-Healthcare record system

7.3. Clinical Decision Support

Clinical decision support features EHRs offer help medical professionals make well-informed choices. These technologies help improve the quality of care and lower medical errors by warning doctors about possible drug interactions, recommending evidence-based treatment alternatives, and flagging aberrant test results.

7.4. Workflow Optimization

By automating repetitive operations like appointment scheduling, billing, and paperwork, EHR systems can improve clinical workflows.

7.5. Telehealth Integration

With the rise of telehealth, Healthcare professionals can now conduct virtual consultations while having real-time access to patient records because of the growing integration of EHRs with telemedicine systems.

7.6. Population Health Management

By combining data from several sources to pinpoint health trends and inequalities within certain populations, EHRs are essential to population health management.

8. Data Flow of Blockchain-Enabled E-Healthcare System

In this framework, as shown in Figure 3, the user provides their details such as name, age, insurance policy number, Aadhar number, contact details, past illness, etc. The hospital or clinic provides details, including the hospital registration number, hospital name, hospital specification, and available facilities for patient care. Additionally, the insurance companies provide the user's policy details. When the user arrives at the hospital in an emergency, the hospital retrieves the user's details through their blockchain ID. This allows the hospital management to access necessary information about the user, supporting the hospital or doctors in diagnosing the illness and making clinical decisions for treatment. The hospital management also retrieves and verifies the user's insurance policy details with the insurance company. Subsequently, the hospital management creates an estimated bill for the user. Upon receiving the estimated bill, the user proceeds to the insurance company desk for approval. The insurance company then evaluates the estimated bill details and approves the status. If the estimate bill status is approved, the hospital management admits the user with the insurance claim process; otherwise, the hospital admits the patient at their own expense, ensuring a streamlined and transparent process for healthcare management and insurance claims within the blockchain framework.

9. Verification of Solidity Contracts in Event-B

Event-B represents a Solidity contract's type declaration, attributes, functions, and other important components. Event-B invariants are traits that are necessary for the contract to operate correctly. As Figure 4 illustrates, these variables are examined using the Rodin platform, which also uses simulation to verify that events are occurring as intended. If the verification result is not true, a developer can find the issue and move on to the modification section to change the smart contracts; if all that checks away may obtain the authentication for the smart contracts and use it.

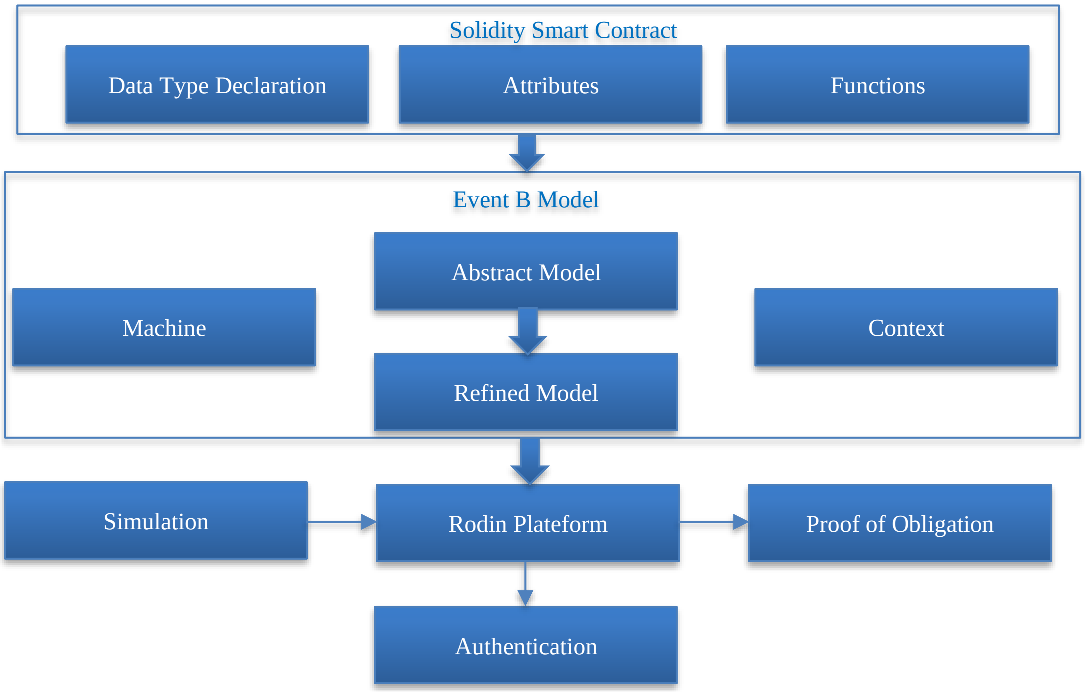


Fig. 4 Model verification of solidity contract in Event-B

10. Formal Verification of Solidity Contract of Blockchain Enabled E-Healthcare System using Event-B

The Event-B model for a Blockchain-Based E Healthcare System contains a context and a machine with multiple events. Guards and activities are present at every event. The invariants specify the characteristics of the variables that shouldn't be broken when the machine's state changes while it is being executed. POLICY, PERSONBLOCKCHAINID, REGISTRATION NO, HOSPITALBLOCKCHAINID, ADHAR, and ESTIMATEDBILL are stated as carrier sets in this Event-B model. The enumerated sets STATUS and BILL STATUS are also defined as shown in Table 1.

Machine Model specifies the dynamic parts of the system. The machine model sees the Context Model and has seven variables and seven invariant properties. It is represented in Table 2.

Table 1. Context of proposed system model

CONTEXT
EHR Context
SETS
PERSONBLOCKCHAINID, ADHAR, POLICY, STATUS, HOSPITALBLOCKCHIANID, REGISTRATIONNO, ESTIMATEDBILL, BILLSTATUS
CONSTANTS
Verified, notverified, approved, notapproved
AXIOMS
axm 1 : partition(STATUS,{verified},{notverified})
axm 2 : partition(BILLSTATUS, {approved}, {notapproved})
END

Table 2. Variable and Invariants of the proposed model

MACHINE
EHR Machine
SEES
EHR Context
VARIABLES
User, UserRegister, RegisterPolicy, Policystatus, Registerhospital, Estimatebill, Estimatebillstatus
INVARIANTS
inv1 : User∈ADHAR↔PERSONBLOCKCHAINID
inv2 : UserRegister∈ADHAR⇸PERSONBLOCKCHAINID
inv3 : RegisterPolicy∈ADHAR⇸POLICY
inv4 : Policystatus∈PERSONBLOCKCHAINID⇸ STATUS
inv5: Registerhospital∈HOSPITALBLOCKCHIANID⇸REGISTRATIONNO
Inv6: Estimatebill∈PERSONBLOCKCHAINID⇸ESTIMATEDBILL
Inv7: Estimatebillstatus∈ESTIMATEDBILL⇸BILLSTATUS

- The invariant inv1 states a relationship between the set of ADHAR (Aadhaar) numbers mapped to their corresponding set of PERSONBLOCKCHAINID. In other words, each user is uniquely associated with an ADHAR number, and that ADHAR number is linked to a specific PERSON BLOCKCHAIN ID.
- The invariant inv2 indicates a relationship between the set of ADHAR numbers mapped to their corresponding PERSON BLOCKCHAIN ID. It means that multiple UserRegister entries can be associated with the ADHAR number, and each ADHAR number is linked to a unique PERSON BLOCKCHAIN ID.
- The invariant inv3 states that there is a relationship between the set of ADHAR numbers mapped to their corresponding POLICY. It means that variable RegisterPolicy entries can be associated with the ADHAR number, but each ADHAR number is linked to a unique POLICY.
- The invariant Inv4 states that the variable Policy status defines the type of policy. The policy can be verified or not verified.
- The invariant Inv5 defines a relationship between the set of HOSPITALBLOCKCHIANID mapped to their corresponding REGISTRATION NO. It means that variable Registerhospital entries can be associated with the HOSPITALBLOCKCHIANID, and HOSPITALBLOCKCHIANID is linked to a unique REGISTRATION NO.
- The invariant Inv6 states that there is a relationship between the set of PERSON BLOCKCHAIN IDs mapped to their corresponding ESTIMATEDBILL. It means that each PERSON BLOCKCHAIN ID is linked to ESTIMATEDBILL entries.
- The invariant Inv7 states that the variable Estimatebillstatus defines the type of Bill status. The bill status can be approved or not approved.

Events are created by converting functions. Require/assert statements are critical in this section as a premise for successfully calling the function and translating them to Event-B guards in the event.

10.1. User Registration

The smart contract code features a function named register person (Algorithm 1) that allows individuals to register by providing their name, age, policy number, Aadhar number, city, email address, and information about past illnesses. The function includes input validation through required statements to ensure that the Aadhar number is equal to 12 digits and the age is greater than zero. Upon successful validation, a new patient record is created and stored in a mapping, and the function returns the message "successfully registered".

Algorithm 1: User Registration

```
Initialize: Input= name, age, policy no, Adharno, City, Mailid, pastillness (Type: string memory, uint, uint, uint, string memory, string memory, string memory Access Specifier: Public returns), Patients, nextpatientId, block.timestamp, msg.sender, P1(Type: Structure), Received (Flag)
for
        name== Received do
        Age== Received do
        policy no== Received do
        Adharno== Received do
        City== Received do
        Mailid== Received do
        pastillness== Received do
    if
        Adharno >= 10**11 && Adharno < 10**12, "Adhar number must have exactly 12 digits"
        Age> 0, "Age should be greater than zero";
    then
        P1<= name, age, policy no, Adharno, City,
        Mailid, pastillness, msg.sender, block.timestamp
        patients[nextpatientId] <= P1
        nextpatientId++
end for
Emit
        return "successfully registered"
```

Add Person ≜	
ANY	
Pbid, a	
WHERE	
grd1 :	$pbid \in PERSONBLOCKCHAINID$
grd2 :	$a \in ADHAR$
grd3 :	$a \mapsto pbid \in User$
grd4 :	$a \notin dom(UserRegister)$
THEN	
act1 :	$UserRegister \coloneqq UserRegister \cup \{a \mapsto pbid\}$
END	

Fig. 5 Formal model of smart contract of user registration

The smart contract representation in the Event-B method for user registration is shown in Figure 5.

The event is named Add person (Figure 5) and represents a regular operation that can occur during system execution. The event has four conditions specified in the WHERE clause: the guard grd1 $pbid \in$ PERSONBLOCKCHAINID; this condition states that the value of "pbid" belongs to the set "PERSONBLOCKCHAINID". It ensures that the value of "pbid" is a valid identifier in the person blockchain.

The guard grd2 $a \in$ ADHAR This condition states that the value of "a" belongs to the set "ADHAR". It ensures that the value of "a" is a valid Aadhaar number. The guard grd3 states that the mapping of "a" to "pbid" exists in the set "User". It ensures that there is a valid mapping between the Aadhaar number "a" and the corresponding person blockchain identifier "pbid" in the set "User".

The guard grd4 $a \in$ dom (UserRegister) This condition states that the value of "a" does not exist in the domain of the set "UserRegister". It ensures that the Aadhaar number "a" is not already registered in the system.

The event has a single action specified act1: UserRegister = UserRegister $\cup\{a \mapsto pbid\}$ This action updates the set "UserRegister" by adding a new mapping between the Aadhaar number "a" and the person blockchain identifier "pbid".

Algorithm 2: Hospital Registration

```
Initialize: Hospitalname, Facility, Department, Registration (Type: string memory, string memory, string memory, uint256, Access Specifier: Public returns) hospitals, nexthospitalid, block.timestamp, msg.sender, H1 (Type: Structure), Received (Flag)
for
        Hospitalname== Received do
        Facility== Received do
        Department== Received do
        Registration== Received do
    if
            Registration> 0, "Age should be greater than zero"
    then
            H1<=Hospitalname, Facility, Department,
            Registration, block.timestamp, msg.sender
            hospitals[nexthospitalid] <= H1
            nexthospitalid++
end for
emit
            return "successfully registered"
```

Addhospital ≜	
ANY	
hbid, r	
WHERE	
grd1 :	$hbid \in HOSPITALBLOCKCHIANID$
grd2 :	$r \in REGISTRATIONNO$
grd3 :	$hbid \notin dom(Registerhospital)$
THEN	
act1 :	$Registerhospital \coloneqq Registerhospital \cup \{hbid \mapsto r\}$
END	

Fig. 6 Formal Model of Smart Contract of Register Hospital

10.2. Hospital Registration

The smart contract code includes a function named register hospital (Algorithm 2) that allows the registration of a hospital by accepting parameters such as the hospital name, facility type, department, and registration details. Upon invocation, the function stores the provided information along with the current timestamp and the sender's address in a mapping called hospitals, using the next hospitalid as the index, then increments the next hospitalid to prepare for the next registration. This enables the seamless storage and organization of hospital registration data within the smart contract, providing a clear record of registered hospitals and their associated details.

The smart contract representation in the Event-B method for Hospital Registration is shown in Figure 6. The event is named Add Hospital (Figure 6) and represents a regular operation that can occur during system execution. The event has two parameters named "hbid" and "r".

These parameters represent the hospital blockchain identifier and the registration number, respectively. The event has three conditions specified. The guard grd1: hbid ∈ HOSPITALBLOCKCHIANID states that the value of "hbid" belongs to the set "HOSPITALBLOCKCHIANID". It ensures that the value of "hbid" is a valid identifier in the hospital blockchain. The guard grd2 states that the value of "r" belongs to the set "REGISTRATIONNO". The guard grd3 hbid ∈ dom (Registerhospital) defines that the value of "hbid" does not exist in the domain of the set "Registerhospital". It ensures that the hospital blockchain identifier "hbid" is not already registered in the system. The event has a single action specified. The action act1 Registerhospital = Registerhospital ∪ {hbid ↦ r} updates the set "Registerhospital" by adding a new mapping between the hospital blockchain identifier "hbid" and the registration number "r".

10.3. Insurance Company Registration

The smart contract module (Algorithm 3), called the RegisterCompany function, is specifically designed for insurance companies.When an insurance company calls this function, they can provide their information and the policy information of the users they cover. The smart contract then stores all of this information in an array dedicated to the insurance company. The purpose of this function is to allow insurance companies to register themselves on the blockchain network and provide their relevant details. By doing so, they can establish trust and transparency in their operations, as all the information they submit is stored immutably on the blockchain.

The smart contract representation in the Event-B method for Insurance Company Registration is shown in Figure 7. The event is named Addpolicy (Figure 7) and represents a regular operation that can occur during system execution. The event has two parameters named "p" and "a". These parameters represent the policy and the Adhar, respectively. The event has four conditions specified. The guard grd1 a ∈ ADHAR states that "a" belongs to the set "ADHAR" and the guard grd2 p ∈ POLICY states that "p" belongs to the set "POLICY".

Algorithm 3: Insurance Company Registration

```
Initialize: Adhar, policy, Amount, Companyname (Type: uint, uint, uint, string memory, Access Specifier Public returns), companies, Company  (Type: Array), Received (Flag)
For
|       Adhar== Received do
|       policy== Received do
|       Amount== Received do
|       Companyname== Received do
|   if
|   |    Adharno >= 10**11 && Adharno < 10**12,
|   |       "Adhar number must have exactly 12 digits"
|   then
|           Company<= Adhar, policy, Amount, Companyname
|           Companies<= Company
end for
```

```
Addpolicy ≙
ANY
    p, a
WHERE
    grd1 : a∈ADHAR
    grd2 : p∈POLICY
    grd3 : a↦p∉(RegisterPolicy)
    grd4 : a∉dom(RegisterPolicy)
THEN
    act1 : RegisterPolicy≔RegisterPolicy∪{a↦p}
END
```

Fig. 7 Formal model of smart contract of register insurance company

The guard grd3 a ↦ p ∉ (RegisterPolicy) defines that the pair "a ↦ p" is not a member of the set "RegisterPolicy". This means that the mapping from "a" to "p" is not already present in the set "RegisterPolicy". The guard grd4 a ∉ dom(RegisterPolicy) states that "a" is not a member of the domain of the set "RegisterPolicy". This means that there is no mapping from "a" to any value in the set "RegisterPolicy". The event has a single action specified. The action act1 RegisterPolicy ≔ RegisterPolicy ∪ {a ↦ p} assigns a new value to the set "RegisterPolicy". It adds the mapping from "a" to "p" to the existing set "RegisterPolicy".

10.4. Retrieval of Patient Information

The smart contract contains a function called getpersondata (Algorithm 4), designed to retrieve and return specific data associated with a patient based on their unique patient ID. The function is public and view, indicating that it can be accessed externally. Upon invocation, the function returns a tuple consisting of the patient's name, age, policy number, Aadhar number, and past illness history. This structured data retrieval mechanism allows for the secure and efficient access of patient information from the blockchain, ensuring that sensitive medical data can be accessed as needed while maintaining the integrity and security of the overall system.

```
Algorithm 4: Retrieval of Patient Information
Initialize: patientId (Type: uint256, (Type: string memory, uint256, uint256,uint256, string memory Access Specifier: Public returns), patientId, patients, policyno, Adharno, pastillness, P1(Type: Structure) Received (Flag)
for
    patientId == Received do
    if
        patientId== patients.P1
    then
        Retrieve Name
        Retrieve Age
        Retrieve policyno
        Retrieve Adharno
        Retrieve pastillness
end for
```

```
GetPersondata  ≙
ANY
    a, result
WHERE
    grd1  :  a∈ADHAR
    grd2  :  result=UserRegister[{a}]
THEN
    skip
END
```

Fig. 8 Formal model of smart contract of retrieval of patient information

The smart contract representation in the Event-B method for Retrieval of Patient Information is shown in Figure 8. The event is named Get Person Data (Figure 8) and represents a regular operation that can occur during system execution.

The event has two parameters named "a" and "result". These parameters represent the Adhar and the result, respectively. The event has four conditions specified as the guard grd1 a ∈ ADHAR states that "a" belongs to the set "ADHAR".

This means that "a" is an element of the set "ADHAR". The guard grd2 result = UserRegister [{a}] states that "result" is equal to the value obtained by looking up the mapping for "a" in the set "UserRegister".

This means that "result" will contain the value associated with "a" in the set "UserRegister". The event has a skip action specified as the skip statement is used to indicate that no action is performed in this event. It represents a null action or a no-output.

10.5. Verification of Policy

The smart contract features a function named verify policy (Algorithm 5), designed to validate whether a patient, identified by their patient ID, has an active insurance policy with any registered companies. Upon invocation, the function iterates through the list of registered insurance companies and compares the patient's policy number with the policy numbers associated with each company. If a match is found, indicating that the patient holds a policy with a registered company, the function returns true. If no matching policy is found after iterating through the companies, the function returns false. This functionality enables the verification of a patient's insurance coverage within the blockchain network, providing a crucial mechanism for validating policy information and facilitating seamless interactions between patients and insurance providers.

The smart contract representation in the Event-B method for Retrieval of Verification of Policy is shown in (Figure 9(a), 9(b)). The given code is written in the Event B method language and defines two operations: Verified Policy (Figure 9(a)) and Not Verified Policy (Figure 9(b)).

The operation Verified Policy (Figure 9(a)) is used to update the status of a policy to "verified" in the system. It has the following parameters: pbid (Person Blockchain ID), p (Policy), and a (Aadhar number).

```
Algorithm 5: Verification of Policy
Initialize: patientId (Type: uint, Access Specifier: Public returns (Type: Bool)), patientId, patients, companies, Received (Flag)

for
        patientId == Received do
    for
        uint i = 0; i < companies.length; i++
        if
            (patients[patientId].policyno==companies[i].policy)
            return true
        else
            return false
    end for
end for
```

```
VerifiedPolicy  ≙
ANY
    pbid, p, a
WHERE
    grd4  :  a↦pbid∈UserRegister
    grd5  :  a↦p∈RegisterPolicy
    grd6  :  pbid∈dom(Policystatus)
THEN
    act1  :  Policystatus(pbid)≔verified
END
```

Fig. 9 (a) Formal Model of smart contract for verified policy

NotVerifiedPolicy ≙
ANY
pbid, p, a
WHERE
grd1 : a↦pbid∈UserRegister
grd2 : a↦p∉RegisterPolicy
THEN
act1 : Policystatus(pbid)≔notverified
END

Fig. 9(b) Formal model of smart contract for not verified policy

The conditions for this operation are as the guard grd1 a ↦ pbid ∈ UserRegister ensures that the pair (a, pbid) exists in the UserRegister relation. In other words, it checks if the given Aadhar number and Person blockchain ID are present or not in the set UserRegister. The guard grd2 a ↦ p ∈ RegisterPolicy checks if the pair (a, p) exists in the set RegisterPolicy. It verifies if the given Aadhar number and policy are present in the set RegisterPolicy. The guard grd3 pbid ∈ dom(Policystatus) ensures that the Person Blockchain ID is a key in the Policystatus relation. It checks if the given Person blockchain ID is present in the domain of the Policystatus relation. The action performed by this operation is act1 Policystatus (pbid) ≔ verified updates the value of Policystatus(pbid) to "verified". It sets the policy status with the given Person blockchain ID to "verified".

The operation Not Verified Policy (Figure 9(b)) is used to update the status of a policy to "not verified" in the system. It has the same parameters as the previous operation: pbid (Person blockchain ID), p (Policy), and a (Aadhar number). The conditions for this operation are as the guard grd1 a ↦ pbid ∈ UserRegister checks if the pair (a, pbid) exists in the UserRegister relation. It verifies if the given Aadhar number and Person blockchain ID are present in the UserRegister relation. The guard grd2 a ↦ p ∉ RegisterPolicy ensures that the pair (a, p) does not exist in the RegisterPolicy relation. It checks if the given Aadhar number and policy are not present in the RegisterPolicy relation. The action performed by this operation is act1 Policystatus (pbid) ≔ notverified updates the value of Policystatus(pbid) to "notverified". It sets the status of the policy with the given Person blockchain ID to "not verified".

10.6. Crete Estimate Bill

The smart contract includes a function called Estimatebill (Algorithm 6), which facilitates the estimation and recording of medical billing details within the blockchain. When invoked, this function appends a new instance of a struct Memo4 to the Estimate array, capturing various parameters related to the billing estimation process. The parameters include the estimated ID, patient's name, Aadhar number, diagnosis, prescribed medicine, charges, the current block's timestamp, and the address of the transaction sender. By leveraging blockchain technology, this function enables the transparent and immutable recording of estimated medical billing information, ensuring the integrity and accessibility of such data for relevant stakeholders within the healthcare ecosystem.

Algorithm 6: Crete Estimate Bill

```
Initialize: _estimateid, _name, _Adharno, _diagnosis, _medicine, _charges ) (Type: uint, string memory, uint256, string memory, string memory, uint256, Access Specifier: Public returns Public), _estimateid, _name, _Adharno, _diagnosis, _medicine, _charges, block.timestamp, msg.sender, Estimate, Memo4 (Type: Array), Received (Flag)
for
        _estimateid== Received do
        _name== Received do
        _Adharno== Received do
        _diagnosis== Received do
        _medicine== Received do
        _charges== Received do
    if
            Adharno >= 10**11 && Adharno < 10**12,
              "Adhar number must have exactly 12 digits"
            charges > 0, " charges should be greater than
              zero";
    then
              Memo4 <= _estimateid, _name, _Adharno,
              _diagnosis, _medicine, _charges,
              block.timestamp, msg.sender
              Estimate <= Memo4
end for
```

Createbill ≙
ANY
a, b, pbid
WHERE
grd1 : a∈ADHAR
grd2 : b∈ESTIMATEDBILL
grd3 : pbid∈PERSONBLOCKCHAINID
grd4 : a↦pbid∈UserRegister
grd5 : pbid∉dom(Estimatebill)
THEN
act1 : Estimatebill≔Estimatebill∪{pbid↦b}
END

Fig. 10 Formal model of smart contract for creating estimate bill

The smart contract representation in the Event B method for the Create Estimate Bill is shown in Figure 10. The given code is written in the Event B method language and defines an operation called Create bill (Figure 10). This operation is used to create a bill for a Person blockchain ID in the system. This operation has the following parameters: a (Aadhar number), b (Estimated bill), and pbid (Person blockchain ID). The conditions for this operation are as the guard grd1 a ∈ ADHAR checks if the given Aadhar number is present in the ADHAR set, and the guard grd2 b ∈ ESTIMATEDBILL checks if the given estimated bill b is present in the ESTIMATEDBILL set. It verifies if the estimated bill is valid. The guard grd3 pbid ∈ PERSONBLOCKCHAINID This condition checks if the given Person Blockchain ID is present in the PERSONBLOCKCHAINID set, and the guard grd4 a ↦ pbid ∈ UserRegister ensures that the pair (a, pbid) exists in the UserRegister relation.

It checks if the given Aadhar number and Person blockchain ID is present in the UserRegister relation; the guard grd5 pbid ∉ dom (Estimatebill) ensures that the Person blockchain ID (pbid) is not already a key in the Estimatebill relation. It checks if the given Person blockchain ID is not present in the domain of the Estimatebill relation. The action performed by this operation act1 Estimatebill ≔ Estimatebill ∪ {pbid ↦ b} action adds a new entry to the Estimatebill relation. It creates a mapping between the Person blockchain ID (pbid) and the estimated bill b, and adds it to the Estimatebill relation.

10.7. Check Estimate Bill Status

The smart contract contains a function called estimatebillstatus (Algorithm 7), which serves the purpose of verifying the billing status for a specific patient identified by their patientId. Upon invocation, the function iterates through the Estimate array to compare the patient's Aadhar number with the Aadhar numbers stored in the billing estimates. If a matching Aadhar number is found, the function returns "verified," indicating that the patient's billing estimate has been recorded.

If no matching Aadhar number is found after iterating through the billing estimates, the function returns "notverified," signifying that the patient's billing estimate has not been recorded. This functionality provides a transparent and secure method for verifying the status of billing estimates within the blockchain, ensuring the integrity and accessibility of billing information for the involved parties.

Algorithm 7: Check Estimate Bill Status

```
Initialize: patientId (Type: uint, Access Specifier: Public
returns (Type: Bool)), patientId, patients, Adharno,
Received (Flag)
for
        patientId== Received do
    for
            uint i = 0; i < Estimate.length; i++
        if
            patients[patientId].Adharno ==
            Estimate[i].Adharno
              return "verified";
        else
              return "notverified";
    end for
end for
```

```
Approvedbill  ≙
ANY
    pbid, a, b
WHERE
    grd1  :  pbid∈PERSONBLOCKCHAINID
    grd2  :  a∈ADHAR
    grd3  :  a↦pbid∈UserRegister
    grd4  :  pbid↦b∈Estimatebill
THEN
    act1  :  Estimatebillstatus(b)≔approved
END
```

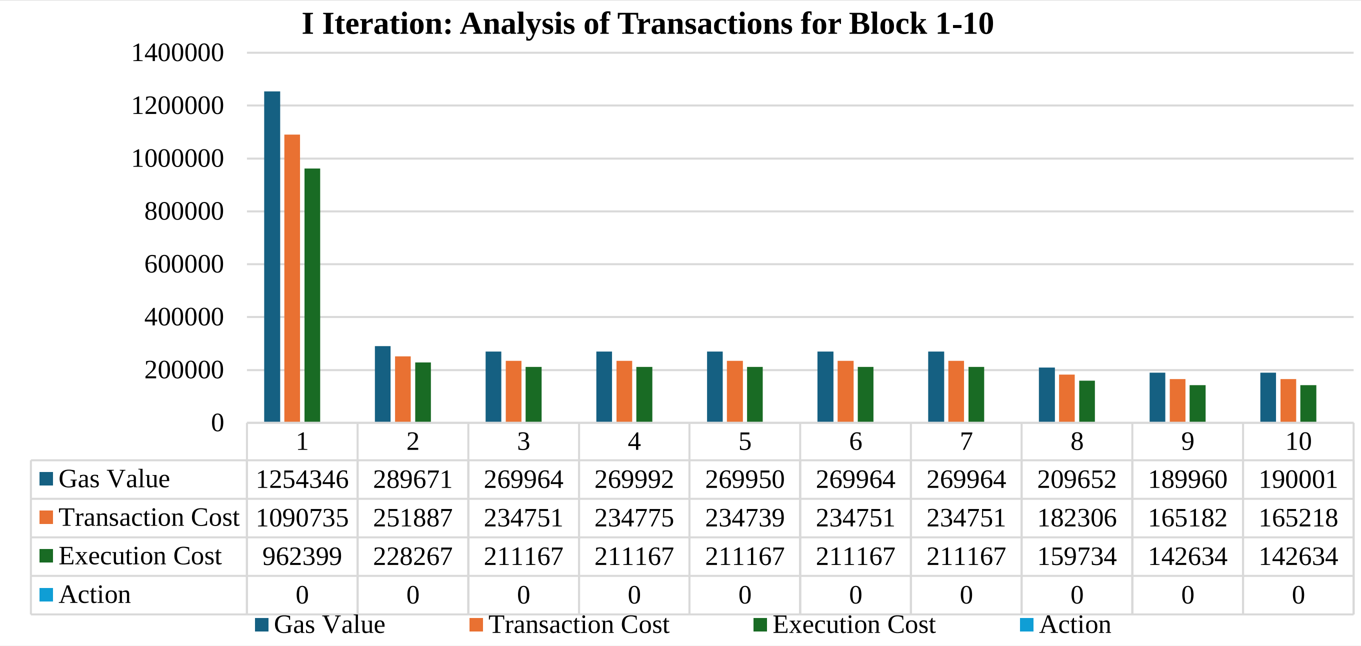
Fig. 11(a) Formal model of smart contract for approved estimate bill

```
NotApprovedbill  ≙
ANY
    pbid, a, b
WHERE
    grd1  :  pbid∈PERSONBLOCKCHAINID
    grd2  :  a∈ADHAR
    grd3  :  a↦pbid∈UserRegister
    grd4  :  pbid↦b∉Estimatebill
THEN
    act1  :  Estimatebillstatus(b)≔notapproved
END
```

Fig. 11(b)Formal model of smart contract for not approved estimate bill

The smart contract representation in the Event-B method for Check Estimate Bill Status is shown in (Figure 11(a), 11(b)). The given code is written in the Event B method language and defines two operations: Approved bill (Figure 11(a)) and Not Approved bill (Figure 11(b)). This operation approves a bill (Figure 11(a)) for a Person blockchain ID in the system. It has the following parameters: pbid (Person blockchain ID), a (Aadhar number), and b (Estimated bill). The conditions for this operation are as the guard grd1 pbid ∈ PERSONBLOCKCHAINID checks if the given Person blockchain ID (pbid) is present in the PERSONBLOCKCHAINID set. The guard grd2 a ∈ ADHAR checks if the given Aadhar number a is present in the ADHAR set. The guard grd3 a ↦ pbid ∈ UserRegister ensures that the pair (a, pbid) exists in the UserRegister relation. It checks if the given Aadhar number and Person blockchain ID are present in the UserRegister relation. The guard grd4 pbid ↦ b ∈ Estimatebill checks if the pair (pbid, b) exists in the Estimatebill relation. It verifies if the given Person blockchain ID and estimated bill are present in the Estimatebill relation. The action act1 Estimatebillstatus(b) ≔ approved updates the value of Estimatebillstatus(b) to "approved". It sets the bill's status with the given estimated bill b to "approved". This operation is used to mark a bill as not approved (Figure 11(b)) for a Person blockchain ID in the system. It has the same parameters as the previous operation: pbid (Person blockchain ID), a (Aadhar number), and b (Estimated bill). The conditions for this operation are as the guard grd1 pbid ∈ PERSONBLOCKCHAINID checks if the given Person blockchain ID (pbid) is present in the

PERSONBLOCKCHAINID set. The guard grd2 a ∈ ADHAR checks if the given Aadhar number a is present in the ADHAR set. The guard grd3 a ↦ pbid ∈ UserRegister ensures that the pair (a, pbid) exists in the UserRegister relation. It checks if the given Aadhar number and Person blockchain ID are present in the UserRegister relation. The guard grd4 pbid ↦ b ∉ Estimatebill ensures that the pair (pbid, b) does not exist in the Estimatebill relation. It checks if the given Person blockchain ID and estimated bill are not present in the Estimatebill relation. The action act1 Estimatebillstatus(b) ≔ notapproved updates the value of Estimatebillstatus(b) to "notapproved". It sets the bill's status with the given estimated bill b to "not approved".

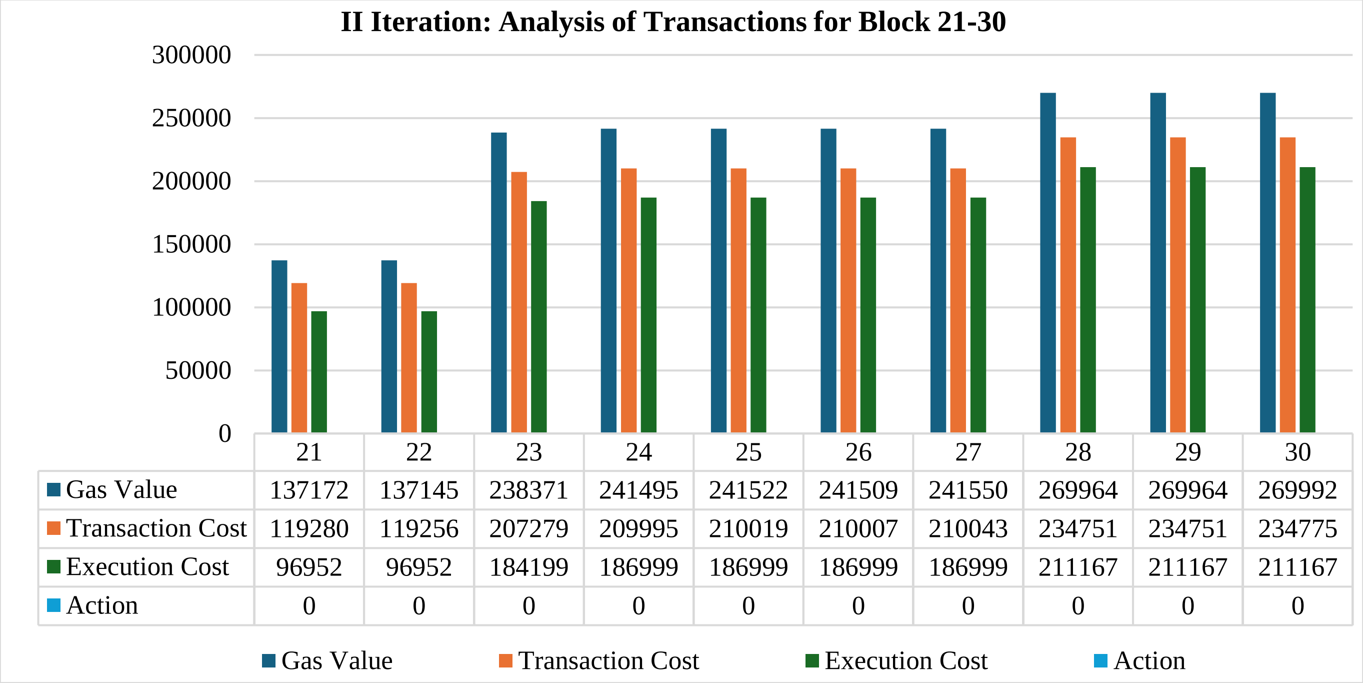


	1	2	3	4	5	6	7	8	9	10
■ Gas Value	1254346	289671	269964	269992	269950	269964	269964	209652	189960	190001
■ Transaction Cost	1090735	251887	234751	234775	234739	234751	234751	182306	165182	165218
■ Execution Cost	962399	228267	211167	211167	211167	211167	211167	159734	142634	142634
■ Action	0	0	0	0	0	0	0	0	0	0

Fig. 12 I Iteration: Analysis of transactions for block 1-10

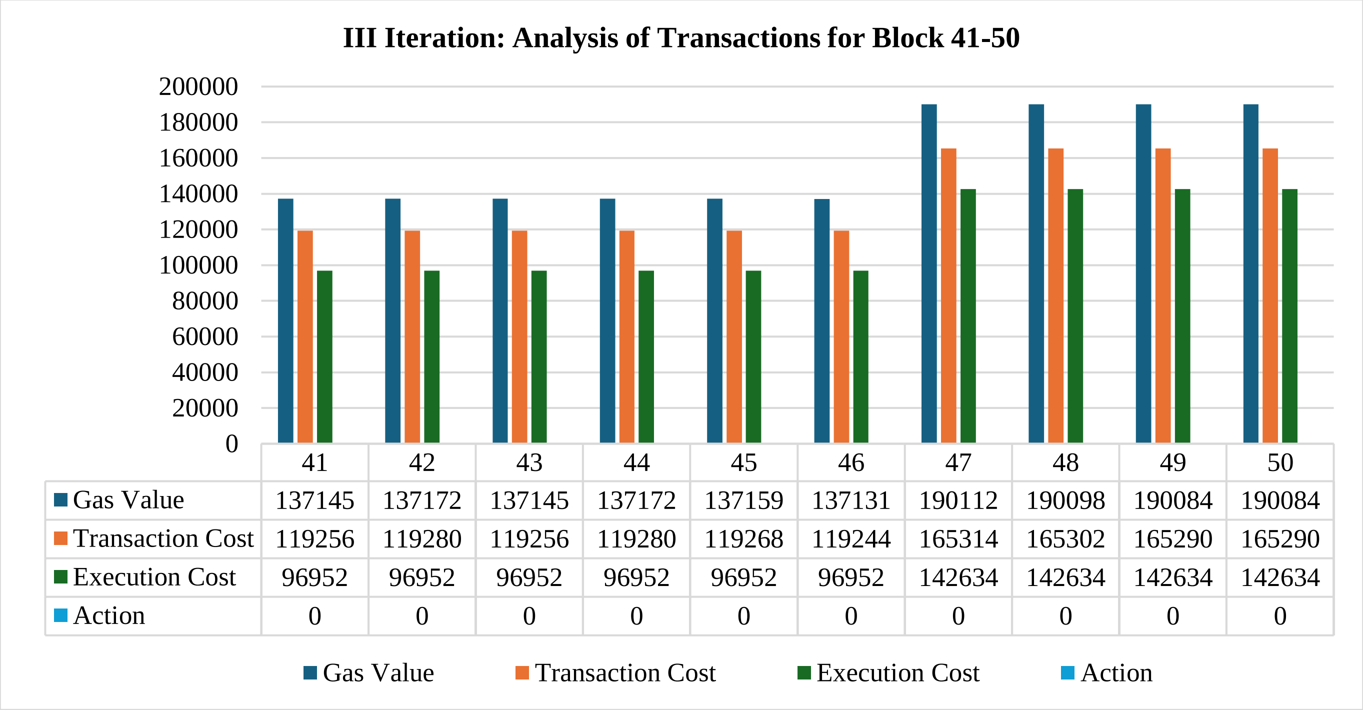
11. Results and Discussion

Blockchain technology offers a range of potential benefits when applied to EHR systems, including data security, privacy, interoperability, and enhanced confidentiality. By storing data on a decentralized network of computers, blockchain makes it significantly more challenging for hackers to access or manipulate sensitive information. In the following implementation mentioned in this paper, the block is not generated, but the gas values are deducted. This happens in the following scenario. Assuming the patient got the policy number from the insurance company. When the patient registers on the blockchain, he fills in all the entries, including the policy number on the blockchain. Blockchain matches the policy number at both places, i.e., with the insurance company and blockchain. The Insurance company also registers itself on the blockchain in the initial stages while considering the Patient's Name, Patient Policy Number, and Policy Amount. While Making An Estimate Bill, Suppose the doctor of any third party officer erroneously enters the incorrect aadhar number of the patient and tries to claim the bill at the insurance company; there will be a mismatch of the aadhar at the doctor or third party office site and an insurance company. The insurance company will not consider this claim as the Adhar numbers are different at both ends. So this ensures the immutability aspect of the blockchain as the data has to be verified at both ends. The graph mentioned above, as shown in Figure 12, highlights the analysis of transactions for blocks 1 to 10. Where 1st block is considered as the geneses block, and the 10th block is the last block that was mined in iteration one. The graph shows the relationship between four variables, namely Block number, Gas Usage, transaction Cost, and Execution Cost. The last variable, action, is basically of string data type; as a result, action highlights the default value of 0. The action for 1st block is Contract Deploy, followed by Register Patient for blocks number 2 to 7 and ending up in a range of 8 to 10 with action Register Hospital. The graph mentioned above, as shown in Figure 13, highlights the analysis of transactions for blocks 21 to 30. In the second iteration, 21 represents the first, and the 30th represents the last block that was mined. It shows the relationship between four variables, namely Block number, Gas Usage, transaction Cost, and Execution Cost. The last variable, action, is basically of string data type; as a result, action highlights the default value of 0. The action for blocks 21st and 22nd is Register Company followed by Estimate Bill for block no. 23rd to 27th and ending up in a range of 28 to 30 with action Register Patient. The graph mentioned above, as shown in Figure 14, highlights the analysis of transactions for blocks 41 to 50. In the third iteration, 41 represents the first, and 50th represents the last block that was mined. It shows the relationship between four variables, namely Block number, Gas Usage, transaction Cost, and Execution Cost. The last variable action is basically of string data type, as a result, action highlights the by default value 0. The action for blocks 41 to 46 is Register Company, followed by Register Hospital for block no. 47 to 50. An Event-B model can be employed to capture the behavior of the Solidity contract. The goal of this model is to appropriately depict the behavior of the Solidity contract. It should encapsulate the same functionality and logic as the original contract. ProB Rodin's plugin is used for function simulation. This confirms that the behaviors of the model match the design, enabling us to approximate the solidity contract with speed. Rodin uses varying values for the model's parameters to simulate the abstract model.



	21	22	23	24	25	26	27	28	29	30
■ Gas Value	137172	137145	238371	241495	241522	241509	241550	269964	269964	269992
■ Transaction Cost	119280	119256	207279	209995	210019	210007	210043	234751	234751	234775
■ Execution Cost	96952	96952	184199	186999	186999	186999	186999	211167	211167	211167
■ Action	0	0	0	0	0	0	0	0	0	0

Fig. 13 II Iteration: Analysis of transactions for block 21-30

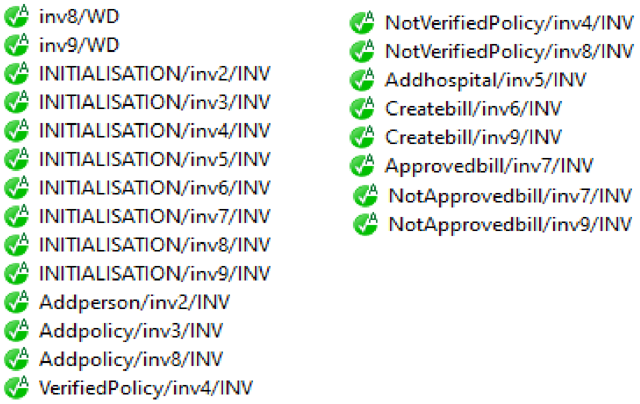


	41	42	43	44	45	46	47	48	49	50
■ Gas Value	137145	137172	137145	137172	137159	137131	190112	190098	190084	190084
■ Transaction Cost	119256	119280	119256	119280	119268	119244	165314	165302	165290	165290
■ Execution Cost	96952	96952	96952	96952	96952	96952	142634	142634	142634	142634
■ Action	0	0	0	0	0	0	0	0	0	0

Fig. 14 III Iteration: Analysis of transactions for block 41-50

This allows us to watch and analyze how the model performs in various settings. The simulation results show that our abstract model executes the required behaviors that are compatible with the original solidity contract. When working with properties, it is customary for each property to produce twenty-two proof obligations, as shown in Figure 15. These proof obligations are essentially statements or conditions that must be proven to establish the property's validity.

Proof obligations are normally proven one by one, which means that each obligation is treated individually. This enables a systematic and rigorous verification process.



inv8/WD
inv9/WD
INITIALISATION/inv2/INV
INITIALISATION/inv3/INV
INITIALISATION/inv4/INV
INITIALISATION/inv5/INV
INITIALISATION/inv6/INV
INITIALISATION/inv7/INV
INITIALISATION/inv8/INV
INITIALISATION/inv9/INV
Addperson/inv2/INV
Addpolicy/inv3/INV
Addpolicy/inv8/INV
VerifiedPolicy/inv4/INV
NotVerifiedPolicy/inv4/INV
NotVerifiedPolicy/inv8/INV
Addhospital/inv5/INV
Createbill/inv6/INV
Createbill/inv9/INV
Approvedbill/inv7/INV
NotApprovedbill/inv7/INV
NotApprovedbill/inv9/INV

Fig. 15 Auto generated proof obligations

- **Inv8:** $\forall pbid,a,p\cdot(p\in POLICY\land pbid\in PERSONBLOCKCHAINID\land a\in ADHAR\land a\mapsto pbid\in User\land pbid\in dom(Policystatus)\land Policystatus(pbid)=verified\Rightarrow a\mapsto p\in RegisterPolicy)$
- The invariant *inv8* states the existence of *p* in the set *POLICY*, *pbid* in the set *PERSONBLOCKCHAINID*, *a* in the set *ADHAR*, and a specific mapping relationship between *a* and *pbid* in the set *User*. Additionally, *pbid* must be a member of the domain of the function *Policystatus*, and the value of *Policystatus(pbid)* must be "*verified*" for the mapping between *a* and *p* to exist in the set *RegisterPolicy*.
- **Inv9:** $\forall a,b,pbid\cdot(a\in ADHAR\land b\in ESTIMATEDBILL\land pbid\in PERSONBLOCKCHAINID\land a\mapsto pbid\in User\land b\in dom(Estimatebillstatus)\land Estimatebillstatus(b)=approved\Rightarrow pbid\mapsto b\in Estimatebill)$
- The invariant *inv9* states that if certain conditions are met, then a mapping between *pbid* and *b* exists in the set *Estimatebill*. The conditions include the existence of *a* in the set *ADHAR*, *b* in the set *ESTIMATEDBILL*, *pbid* in the set *PERSONBLOCKCHAINID*, and a specific mapping relationship between *a* and *pbid* in the set *User*. Additionally, *b* must be a member of the domain of the function *Estimatebillstatus*, and the value of *Estimatebillstatus(b)* must be "*approved*" for the mapping between *pbid* and *b* to exist in the set *Estimatebill.*

11.1. Comparative Analysis

The work presented in this paper [25] focuses on permissioned blockchain, while this paper examines consortium blockchain, which offers the additional advantage of maintaining privacy or public access according to specific requirements. Furthermore, this paper incorporates the Event B method using the Rodin tool to develop a formal model for verifying a blockchain-enabled EHR system, enhancing the reliability and correctness of the EHR system. This formal approach not only ensures that the system adheres to specified requirements but also facilitates the identification of potential vulnerabilities and inconsistencies within the EHR framework. By leveraging the strengths of consortium blockchain, this paper aims to create a more robust and secure environment for managing sensitive health data, ultimately improving patient trust and data integrity. In contrast, paper [25] does not significantly contribute to the application of the Event B method, which limits its ability to address the formal verification aspects that are crucial for the deployment of secure and reliable EHR systems in healthcare settings. This distinction highlights the innovative approach taken in this paper.

12. Conclusion and Future Scope

The suggested approach in this research intends to improve the accuracy of EHR System Solidity contracts by converting them to the Event-B formal modeling language and using formal verification to validate their features. Our method specifies the EHR system functionality from Solidity languages to the Event-B model while preserving the semantics of the original contracts. To demonstrate the applicability of our technique, we effectively transformed an EHR System Solidity contract into the Event-B model using our translation. Furthermore, the Rodin tool performs basic property verification and type checking automatically. Given the significance of securing Solidity contracts involving large transactions, we feel that adapting our technique to other contracts in this domain would be extremely advantageous. The method presented in this paper uses the Event B method to enable functional verification of EHR System Solidity contracts at different levels of abstraction. We can verify sophisticated properties of Solidity contracts using the highly expressive language of first-order logic. We intend to expand the present Solidity contracts to include more types and capabilities as part of our future work. This enhancement will provide a more extensive definition of Solidity contracts, allowing for the verification of more complex features.

Acknowledgments

This work is done under the “Formal Development and Verification of Secure Blockchain-Enabled Digital Healthcare System using Event-B” project governed by the Uttar Pradesh Council of Science and Technology (UPCST) and supported by Pranveer Singh Institute of Technology, Kanpur, Uttar Pradesh, India.

References

[1] Asma Khatoon, “A Blockchain-based Smart Contract System for Healthcare Management,” *Electronics*, vol. 9, no. 1, pp. 1-23, 2020. [CrossRef] [Google Scholar] [Publisher Link]

[2] M. Sumathi et al., “Appointment Booking and Drug Inventory System in Healthcare Services using Blockchain Technology,” *ADCAIJ: Advances in Distributed Computing and Artificial Intelligence Journal*, vol. 12, no. 1, pp. 1-16, 2023. [CrossRef] [Google Scholar] [Publisher Link]

[3] Hengrui Jia et al., “Proof-of-Learning: Definitions and Practice,” *IEEE Symposium on Security and Privacy (SP)*, San Francisco, CA, USA, pp. 1039-1056, 2021. [CrossRef] [Google Scholar] [Publisher Link]

[4] John von Neumann, *The General and Logical Theory of Automata*, Systems Research for Behavioral Science, Routledge, pp. 97-107, 2017. [Google Scholar] [Publisher Link]

[5] Shahid Khan, Osman Hasan, and Atif Mashkoor, “Formal Verification and Safety Assessment of a Hemodialysis Machine,” *International Conference on Current Trends in Theory and Practice of Informatics*, Springer International Publishing, pp. 241-254, 2017. [CrossRef] [Google Scholar] [Publisher Link]

[6] Sulskus Gintautas, “*An Investigation into Event-B Methodologies and Timing Constraint Modeling*,” Doctoral Dissertation, University of Southampton, pp. 1-217, 2017. [Google Scholar] [Publisher Link]

[7] Racem Bougacha et al., “Extending SysML with Refinement and Decomposition Mechanisms to Generate Event-b Specifications,” *International Symposium on Theoretical Aspects of Software Engineering*, Springer International Publishing, pp. 256-273, 2022. [CrossRef] [Google Scholar] [Publisher Link]

[8] Ariel Ekblaw et al., “A Case Study for Blockchain in Healthcare: “MedRec” Prototype for Electronic Health Records and Medical Research Data,” *Proceedings of IEEE Open & Big Data Conference*, vol. 13, 2016. [Google Scholar]

[9] Sabita Khatri et al., “A Systematic Analysis on Blockchain Integration with Healthcare Domain: Scope and Challenges,” *IEEE Access*, vol. 9, pp. 84666-84687, 2021. [CrossRef] [Google Scholar] [Publisher Link]

[10] Mohan Kubendiran, Satyapal Singh, and Arun Kumar Sangaiah, “Enhanced Security Framework for E-Health Systems using Blockchain,” *Journal of Information Processing Systems*, vol. 15, no. 2, pp. 239-250, 2019. [CrossRef] [Google Scholar] [Publisher Link]

[11] A.G. Chandini, and P.I. Basarkod, “Patient Centric Pre-Transaction Signature Verification Assisted Smart Contract based Blockchain for Electronic Healthcare Records,” *Journal of Ambient Intelligence and Humanized Computing*, vol. 14, no. 4, pp. 4221-4235, 2023. [CrossRef] [Google Scholar] [Publisher Link]

[12] Mouhamad Almakhour et al., “Verification of Smart Contracts: A Survey,” *Pervasive and Mobile Computing*, vol. 67, 2020. [CrossRef] [Google Scholar] [Publisher Link]

[13] Palina Tolmach et al., “A Survey of Smart Contract Formal Specification and Verification,” *ACM Computing Surveys (CSUR)*, vol. 54, no. 7, pp. 1-38. 2021. [CrossRef] [Google Scholar] [Publisher Link]

[14] Yuepeng Wang et al., “Formal Verification of Workflow Policies for Smart Contracts in Azure Blockchain,” *Verified Software. Theories, Tools, and Experiments: 11th International Conference*, New York City, NY, USA, pp. 87-106, 2020. [CrossRef] [Google Scholar] [Publisher Link]

[15] Ikram Garfatta et al., “A Survey on Formal Verification for Solidity Smart Contracts,” *Proceedings of the Australasian Computer Science Week Multiconference*, pp. 1-10, 2021. [CrossRef] [Google Scholar] [Publisher Link]

[16] Leonardo Alt, and Christian Reitwiessner, “SMT-based Verification of Solidity Smart Contracts,” *Leveraging Applications of Formal Methods, Verification and Validation. Industrial Practice: 8th International Symposium, ISoLA, Limassol, Cyprus*, pp. 376-388, 2018. [CrossRef] [Google Scholar] [Publisher Link]

[17] Anu Raj, and Shiva Prakash, “Smart Contract-based Secure Decentralized Smart Healthcare System,” *International Journal of Software Innovation (IJSI)*, vol. 11, no. 1, pp. 1-20, 2023. [CrossRef] [Google Scholar] [Publisher Link]

[18] Abid Hassan et al., “Secured Insurance Framework using Blockchain and Smart Contract,” *Scientific Programming*, vol. 2021, pp. 1-11, 2021. [CrossRef] [Google Scholar] [Publisher Link]

[19] Sanjeev Kumar Dwivedi et al., “Blockchain-based Electronic Medical Records System with Smart Contract and Consensus Algorithm in Cloud Environment,” *Security and Communication Networks*, vol. 2022, pp. 1-10, 2022. [CrossRef] [Google Scholar] [Publisher Link]

[20] Eman H. Alkhammash, “Trustworthy Smart City Systems using Refinement and Event-B Theories,” *Multimedia Tools and Applications*, vol. 81, no. 1, pp. 615-636, 2022. [CrossRef] [Google Scholar] [Publisher Link]

[21] Guillaume Dupont et al., “Event-B Hybridation: A Proof and Refinement-based Framework for Modeling Hybrid Systems,” *ACM Transactions on Embedded Computing Systems (TECS)*, vol. 20, no. 4, pp. 1-37, 2021. [CrossRef] [Google Scholar] [Publisher Link]

[22] Xinpeng Shen et al., “A Novel Method for Causal Structure Discovery from EHR Data and its Application to Type-2 Diabetes Mellitus,” *Scientific Reports*, vol. 11, no. 1, pp. 1-9, 2021. [CrossRef] [Google Scholar] [Publisher Link]

[23] Ravi Sharma et al., “Design of Blockchain-based Precision Health-Care using Soft Systems Methodology,” *Industrial Management & Data Systems*, vol. 120, no. 3, pp. 608-632, 2020. [CrossRef] [Google Scholar] [Publisher Link]

[24] Na Hong et al., “Integrating Structured and Unstructured EHR Data using an FHIR-based Type System: A Case Study with Medication Data,” *AMIA Summits on Translational Science Proceedings*, vol. 2018, pp. 74-83, 2018. [Google Scholar] [Publisher Link]

[25] James Kolapo Oladele et al., “A Blockchain Electronic Health Data System for Secure Medical Records Exchange,” *Journal of Computing Theories and Applications*, vol. 1, no. 3, pp. 231-242, 2024. [CrossRef] [Google Scholar] [Publisher Link]