HOW TO EXTEND THE SECURE TROPOS METHODOLOGY TO BUILD BLOCKCHAIN-BASED SYSTEMS (BBS)?

M2 IKSEM Sorbonne

2023-2024

Master Thesis

Drafted and presented by

SOW Marieme Soda

Professional Tutor within the organization: BUDACA Sandrut Gabriel

Teaching tutor: HERBAUT Nicolas

Jury members: HERBAUT Nicolas, SALINESI Camille

Abstract:

Blockchain is an emerging technology that has gained popularity in various domains due to its robust security features. As more people are now resorting to blockchain, projects about it are fast increasing, even though designing those systems still remains challenging. In fact, existing modelling techniques are not customized for BBS as they do not offer enough options for engineers to design appropriate models. To meet this gap, a new field known as Blockchain Oriented Software Engineering (BOSE) has been created to provide well-defined aspects and best practices in blockchain systems design and modelling. However, such an approach does not propose a standardized solution for BBS modelling, thus emphasizing the need for a specialized methodology.

Though very few researches attempt at launching BBS relevant methodologies, they are not yet mature and do not focus on security.

To bridge this gap, this paper mainly highlights how to extend the Secure Tropos methodology to support BBS while enabling developers and software engineers to fully address security concerns in BBS development process. The methodology was validated on the BC24-Trace case study, a traceability system using blockchain.

Key words: Security, Secure Tropos, Blockchain Modelling, Blockchain-Oriented Software Engineering,

Table of Contents

Table of figures

Table of tables

Acknowledgements

I would like to express my deepest gratitude to all for their warm welcome, availability, exceptional collaboration and support. Their expertise, enthusiasm, and willingness to lend a helping hand have been invaluable. Their insights and contributions have significantly enriched my learning experience and my present master thesis.

I would also like to extend my thanks and appreciation to the entire teaching team at Paris 1 Pantheon Sorbonne University and IKSEM professional trainers for their expertise, mentorship and for creating a conducive learning environment.

I would also like to express my thanks and appreciation to Mr. Nicolas HERBAUT, computer science researcher and head of Master 1 MIAGE at Paris 1 Pantheon Sorbonne University, my teaching tutor, for his availability, willingness to share, constructive feedback and active support as part of his technical and pedagogic monitoring mission.

I would like to extend a special “thank you” to my Colleagues at BNP Paribas, particularly my internship supervisor, Mr. Sandrut Gabriel BUDACA, Security Architect at Network Security. His expertise, trust in my abilities and constant encouragement have been exceptional in my growth and development during these two years of internship. I am thankful for the opportunities I was offered to work on meaningful projects and be exposed to real-world challenges and industry best practices.

I would finally like to extend my warmest wishes and gratitude to my dear parents (my beloved Dad in particular), my relatives, friends, classmates who have highly contributed to my own life and skill development. Their love, patience, insights, feedback, sustained support and encouragement have motivated me to push beyond my limits and always strive for excellence.

Thanks so much everyone and best regards.

List of acronyms

A: Anonymity

AC: Access Control

BBA: Blockchain-Based Applications

BBS: Blockchain-Based Systems

BC: Blockchain

BOS: Blockchain-Oriented Software

BOSE: Blockchain-Oriented Software Engineering

C: Confidentiality

CITES: Convention on International Trade in Endangered Species of Wild Fauna and Flora

CS3D: Corporate Sustainability Due Diligence Directive

DLT: Distributed Ledger Technology

EU: European Union

ER: Entity-Relationship

GDPR: General Data Protection Regulation

GPS: Global Positioning System

GPIO: General Purpose Input/Output

HE: Hardware Event

IoT: Internet of Things

ISO: International Organization for Standardization

MBT: Model-Based Testing

MDE: Model-Driven Engineering

N/A: Not Available

NFC: Near-Field Communication

NFT: Non-Fungible Token

NHE: Non-Hardware Event

PA: Privacy Accountability

P2P: Peer-to-Peer

PESTOS: Privacy Enhanced Secure Tropos

PET: Privacy-Enhancing Technologies

POC: Proof of Concept

RFID: Radio Frequency Identification

SCM: Supply Chain Management

SCQIS: Blockchain-enabled Supply Chain Quality Information Systems

SLR: Systematic Literature Review

SME: Small or Medium-sized Enterprise

UML: Unified Modelling Language

UN: United Nations

I. INTRODUCTION

In recent years, blockchain has emerged as a foundational technology with the potential to revolutionize data integrity and the way we record and verify transactions. Much like the Internet transformed data access and communication, blockchain promises a similar level of evolution by ensuring the immutability and transparency of data. In fact, blockchain operates as a time-stamped, append-only series of blocks of immutable data that cannot be altered. Its promise to drive new business possibilities is evident with its rapid growth and adoption in various sectors, including finance, supply chain management, and healthcare, which it has already transformed.

However, alongside the benefits of blockchain, many security challenges have emerged, particularly concerning smart contracts. Incidents such as the Mt. Gox hack and the DAO attack have underscored the vulnerabilities inherent in current blockchain implementations. These incidents resulted in substantial financial losses and have been attributed to poor software development practices. Moreover, with the rapid development and deployment of blockchain-based systems, we need to implement robust and secure design methodologies.

To achieve this, we could utilize existing practices in software engineering. However, the effectiveness of various software development methodologies has been a long-standing debate within the software engineering community, as no single approach is perfectly suited to every type of software development setting [1]. This is particularly true for blockchain-based systems, which are relatively new and have their own unique requirements. Therefore, it is important to increase the efficiency of software implementation by adopting software process standards.

In this context, BOSE has emerged as a discipline aimed at addressing these challenges by integrating traditional software engineering principles with the unique requirements of blockchain technology. BOSE seeks to provide systematic, disciplined, and quantifiable approaches to blockchain development [1].

Despite the progress in BOSE, there remains a lack of standardized methodologies tailored for the security needs of blockchain systems. The Secure Tropos methodology, known for integrating security considerations into the system development lifecycle, offers a promising foundation for addressing these challenges. By extending Secure Tropos, we can develop a comprehensive framework for modelling and securing blockchain-based systems. This is why this paper proposes an extension of the Secure Tropos methodology to build blockchain-based software systems and, in turn, help stakeholders elicit their security, design, development, and test requirements. The proposed methodology is validated through a case study on BC24-Trace, a blockchain-based traceability system.

For the convenience of this thesis and to facilitate understanding, it will be structured into eight (8) major sections:

- Section 1, provides a general introduction, setting the context and motivation for the thesis.
- Section 2 gives an overview of the concepts used in the thesis (Secure Tropos, blockchain, etc.) and presents the related works.
- Section 3 outlines the research methodology adopted to arrive to this thesis.
- Section 4 examines Secure Tropos concepts to determine how blockchain concepts are overlooked and explains the reasons for extending Tropos.
- Section 5, details the integration of blockchain-specific concepts into the Secure Tropos methodology, including new modelling constructs and their application.

- Section 6, presents the case study of BC24-Trace, illustrating how the proposed methodology can be applied to real-world scenarios to model and secure blockchain systems.
- Section 7 discusses the effectiveness of the proposed methodology and presents how it helps towards the development of secure blockchain systems.

- Section 8 summarizes the findings, responds to research questions, and presents limitations and directions for future work.

Sources and references will be specified throughout the thesis.

II. BACKGROUND

1. Blockchain

i. What is blockchain?

Many definitions of blockchain can be recorded throughout existing literature:

a) according to [2], a peer-to-peer distributed ledger (P2P DLT) requiring a block architecture where each new block is linked to the previous one, and which can be combined with a data model and a communication language enabling smart contracts and other assisting technologies;

b) a "P2P distributed ledger that is cryptographically secure, append only, immutable (extremely hard to change) and updatable only via consensus or agreement among peers" [3];

c) a distributed ledger where each node holds a replica of the ledger;

d) a new paradigm for trust in decentralized systems, facilitating P2P transactions without involving any central role or third party (from a business perspective) …

But regardless of these different perceptions, whether from a business or technical viewpoint, it remains obvious that a blockchain can be basically defined as "a combination of ideas and concepts from many technologies (e.g. electronic cash schemes and distributed systems)" [3]. (See Figure 1)

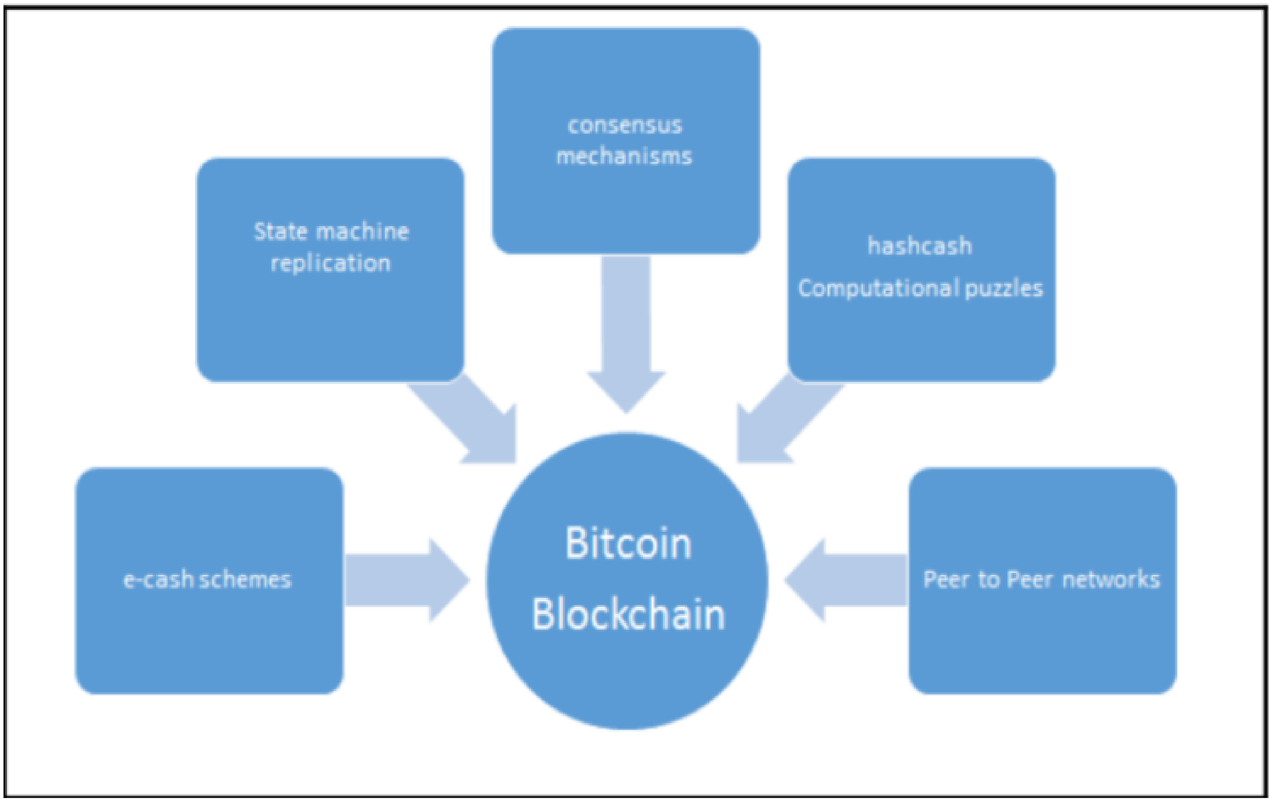


Figure 1: Concepts and ideas that helped with the invention of Bitcoin and Blockchain, from [3].

ii. Blockchain emergence and development

The blockchain system was first introduced with the invention of Bitcoin in 2008, presented in the paper "Bitcoin: A Peer-to-Peer Electronic Cash System" [4], written under the pseudonym Satoshi Nakamoto.

As mentioned earlier, blockchain is a P2P DLT structured as a chain of blocks. To form this chain, blocks are accumulated. In fact, a node initiates a transaction by signing it with its private key. Then this transaction is propagated to peers for validation, usually using the gossip protocol. Once validated by multiple nodes, based on predefined criteria, the transaction is included in a block, which is then sent across the network. The newly created block is cryptographically linked to the previous one and becomes now part of the blockchain.
Over time, it has been observed that the development of blockchain can be organized into 3 generations as follows:

- ***Blockchain 1.0: digital currency:*** This first generation of blockchain is basically used for cryptocurrencies (e.g., bitcoin or digital coin) [3]. This decentralized system allows participants to make transactions directly without the need for a central authority. However, the problem with this generation is related to the energy consumption of consensus algorithms and the fact that proof-of-work gives rewards to the participants who already have the most computation power, which can be a security issue [5].

- ***Blockchain 2.0: smart contract:*** This generation introduced the concept of smart contract to support the self-execution of programs that execute when specific terms are met. The smart contract code is stored on a public ledger and cannot be changed. This second-generation blockchains are used for financial services. They are limited in terms of performance and scalability [5].

- ***Blockchain 3.0: scalability:*** The third generation of blockchain focuses on improving the scalability identified in previous generations [5]. They are used for applications beyond the financial services industry [3]. They have a variety of applications, such as e-Health and supply chain management systems.

Generally, Blockchains can be divided into three types depending on data availability and access mechanisms of their nodes: public Blockchain, consortium Blockchain, and private Blockchain.

- ***Public blockchain***: This is a permissionless ledger accessible to everyone on the network, where anyone can view, read, and write data. Examples include Ethereum and bitcoin.

- ***Private blockchain***: This type is permissioned, meaning only specific individuals can verify and add transaction blocks. Examples are Monax and Multichain.

- ***Consortium blockchain***: Also known as federated or public permission blockchain, this type allows only a group of organizations to verify and add data. It can be open to all or restricted to a particular group. R3 and Corda are examples of Consortium blockchain.

iii. Blockchain’s components

Blockchain systems include a range of core components explained below:

- ***Transaction:*** the core unit of blockchain. It involves exchange or transfer of digital assets (information, goods, services, funds or set of rules which can trigger another transaction) from one address to another. Transactions include information such as transaction time, transaction amount, recipient address, etc.
- ***Block:*** composed of several transactions and a block header that contains previous block hash, merkle tree, nounce, timestamp… (See Figure 2)

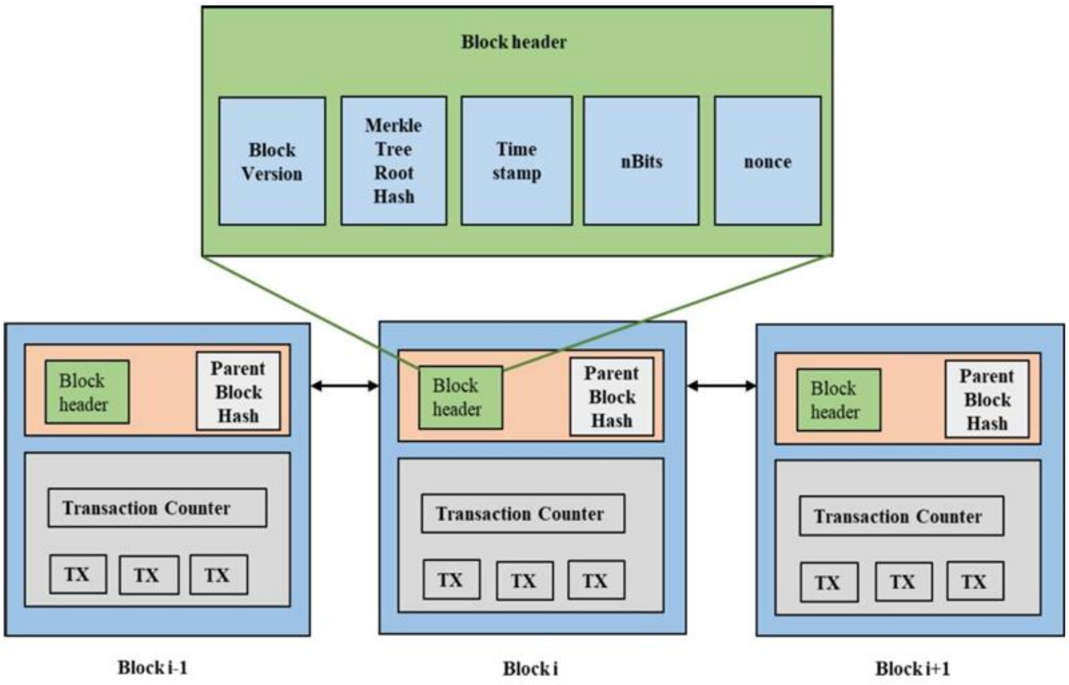


Figure 2: Block components, from [6].

- ***Node:*** Blockchain nodes are computing devices connected to the blockchain that support the network by maintaining a copy of the ledger [2].

There are two (2) types of blockchain nodes:

  - ***Full nodes:*** these nodes store records replicas and verify blockchain data integrity [2].
  - ***Lightweight nodes*:** They do not download the whole ledger, but just a subset of it, when connected to blockchain. These nodes rely on full nodes to facilitate their interaction with the network, enabling them to broadcast their transactions. Typically, lightweight nodes download block headers to verify if a transaction has been incorporated into a block, without the need to store the entire transaction history [2].

- ***Smart contracts:*** A smart contract is a computer program stored in the blockchain and which contains some business logic likely to execute when certain conditions within the system are met. Not all blockchains can implement smart contracts (e.g. blockchain 1.0).

2. Blockchain-Based Systems and Blockchain Oriented Software Engineering

Blockchain-Based Applications (BBAs) are software applications that run on top of a blockchain platform [7]. Unlike traditional applications, their development can be challenging due to their unique characteristics. They are functioning in a decentralized manner where each party adheres to predefined rules without the involvement of any third party.

Given the specialized nature of these applications, particularly the blockchain technology they rely on, software engineers could benefit from the application of BOS-specific software engineering practices. Recognizing this need, some researchers [8, 9, 10] have introduced BOSE as a specialized field of software engineering dedicated to developing software systems that utilize blockchain technology. BOSE major goal is to enhance the quality and efficiency of software regarding blockchain technology.

In this context, the research discussed in paper [8] outlines challenges encountered in the development of blockchain-based software projects. These challenges include security issues, shortcomings in software development practices which may result in design flaws and security vulnerabilities, thereby compromising the quality of blockchain software. Additionally, difficulties arise in smart contract development concerning security, testing, and validation. In response to these challenges, the authors advocate for the introduction of new professional roles that integrate expertise in finance and law. They also propose addressing security concerns through the adoption of methodologies such as Cleanroom SE, software reviews, smart contracts testing, and blockchain transaction testing [8]. Furthermore, they suggest the development of new modelling languages and specialized metrics to assess resource consumption, communication capability, complexity, and performance of blockchain-enabled systems.

Further support for BOSE comes from paper [9], where authors highlight the importance of defining design patterns and anti-patterns and adopting robust testing techniques to mitigate vulnerabilities in smart contracts. They also suggest a combination of manual and automated testing methods, including code coverage approaches, model-based testing (MBT), and formal verification techniques to analyse assumptions and identify faults in smart contracts.

Similarly, paper [10] underscores the significance of smart contract testing for bug and vulnerability detection, emphasizing the need to include other test types like integration and regression, which have been somewhat neglected in existing literature. The authors propose establishing a catalog of attack detection patterns specific to smart contracts and developing a systematic methodology comprising practices, tools, and procedures to enhance smart contract security.

From what we have seen here, BBS development have several areas that needed improvement. Security concerns remain prevalent, with existing practices inadequately addressing vulnerabilities in smart contracts. Additionally, the absence of standardized methodologies and the poor integration of legal and regulatory requirements also pose problems. Even though efforts have been made to tackle some of these issues, the proposed approaches are still far from being complete. For these reasons, research highlights the urgent need for specialized modelling tools and methods to support BOSE effectively. This includes developing new modelling languages and best practices, as well as testing approaches.

3. Secure Tropos

Given the gaps identified above, it is important to identify the tools that can help addressing them or even consider creating our own language. In our case, we have chosen Secure Tropos because it is already well-known in the literature and has proven his effective security modelling. It covers the entire development process, from early requirements to design, and integrates security considerations throughout the development cycle. Even though it is not complete, it has the potential to be extended to fit our needs.
Secure Tropos is based on the Tropos methodology. To fully understand Secure Tropos, it is essential to first present an overview of the Tropos Methodology, as it provides the fundamental principles and processes on which Secure Tropos is built, and therefore enables a clear understanding of Secure Tropos.

i. Tropos methodology

Tropos is an agent-oriented methodology which supports the entire software development process, from analysis to implementation. It involves creating and refining a model of the system and its environment, which helps in development, documentation, and software evolution. The Tropos approach (also in Secure Tropos) supports the system development through five (5) phases: early and late requirements and architectural, detailed design and implementation. Here is how the creator presented the four (4) first phases:

<u>Early requirements analysis:</u>

This phase concerns the understanding of a problem. Indeed, the environment is analysed in terms of its socio and organizational setting. The output of this phase is an organisational model, which includes relevant actors, their respective dependencies and the security constraints imposed to those actors [13]. At this point, we are not interested in describing the system-to-be, it will be done in the next step.

Some questions to guide the analysis have been proposed by author of [14]:

- Who are the main actors?
- What are their goals?
- How can they achieve them?
- Does an actor depend on another one to achieve its goals?

By taking the perspective of each single actor's goal, other questions can be asked:
- How can it be declined into sub-goals?
- Are there alternative ways to reach a goal?
- How to evaluate/choose an alternative?
- Are there means (e.g. plans/resources) to achieve them?

<u>Late requirements analysis:</u>

It defines the system-to-be in the context of its operational environment and along with relevant functions and qualities [12]. A new participant (actor), representing the future system, is introduced, and its dependencies with other actors are analysed. In fact, existing actors assign responsibilities to the system for certain goals and dependencies they cannot fulfil. Those delegated dependencies help identify both system's functional and non-functional requirements [13]. The process involves goals decomposition, means-end, and contribution analysis on the system's goals [14].

<u>**Architectural design**</u>

This part deals with the definition of system architecture. Developers need to choose a system design that can satisfy, as much as possible, the system's security requirements. This involves the accomplishment of three (3) activities:

1) Decomposing and refining the system actor diagram [14],
   - definition of the overall architecture;
   - inclusion of new actors due to delegation of subgoals upon goal analysis of system's goals;
   - inclusion of new actors according to the choice of a specific architectural style (design patterns);
   - inclusion of new actors contributing positively to the fulfilment of some non-functional requirements;
2) Identifying actors' capabilities needed to fulfil their goals [14],
3) Agent assignment, in which a set of agent types is defined and each agent is assigned one or more capabilities [13].

At the end of the process, the design of the actor system is completed, with sub-actors being introduced and assigned sub-goals or tasks to meet the system's objectives. This stage also involves the identification of agents and the definition of their specific capabilities. From a security perspective, it is crucial to identify the security constraints and secure entities introduced by the new actors. Then, during the decomposition process, security sub-constraints and sub-entities are further identified. Finally, each agent in the system is assigned secure capabilities to ensure that all security requirements are effectively addressed [13].

<u>**Detailed design**</u>

This step specifies each architectural component in terms of inputs, outputs, controls and other relevant information [12]. Capabilities, plans and agents' interactions are specified in detail by taking into account the security aspects derived from the previous steps of the analysis. UML notation is used, and it includes the introduction of "security rules," which is a security version of business rules found in UML [13].

Now we have presented the Tropos methodology, we can move to Secure Tropos.

ii. Secure Tropos

Mouratidis [11], introduced Secure Tropos as an extension of the agent-oriented methodology Tropos, with a focus on integrating security into system development. Serving as a security requirement engineering methodology, it considers security issues throughout the whole development process [11]. Key aspects of the approach include early analysis of social security issues, security evaluation simultaneously with other system requirements, and in-depth examination of security aspects during the system design phases [12]. The language uses security concepts such as security constraint, secure goal, secure plan, secure resource, and threat to capture the security requirements from both social and organisational settings. Various activities contribute to the requirements capturing and the analysis of a multi agent system, beginning from early requirement model to its refinement and evolution into subsequent models. Those activities are:

Actor modelling: consists of identifying and analysing the actors in both the environment and the system, including agents. In the early requirement phase, actor modelling focuses on modelling stakeholders' application domains and their intentions as social actors who want to achieve goals. During the late requirement phase, it defines a new actor to represent the system-to-be, determining its functional and non-functional requirements. In architectural design, actor modelling structures the system into subsystems, connected through data and control flows. In detailed design, the system's agents are defined with all necessary details for the implementation platform [11].

Dependency Modelling: Actors in a system often rely on each other to achieve certain goals. To understand these dependencies, Tropos uses dependency modelling across its early and late requirements analysis stages, as well as architectural design. During the early requirements analysis stage, it identifies dependencies between actors in the system's organizational setting [11]. In late requirements analysis stage, dependency modelling focuses on analysing the dependencies of the system-to-be actor, some modifications to the diagram from the previous phase may be needed because of the system's introduction. Finally, in architectural design, data and control flows between sub-actors of the system-to-be actors are modelled in terms of dependencies, providing the basis for the capability modelling that will start later in architectural design together with the mapping of system actors to agents [15].

Goal Modelling: consists of analysing an actor's goal in order to have a more detailed and precise definition of that actor. The diagram is created by using three (3) basic reasoning techniques: means-end analysis, contribution analysis, and AND/OR decomposition. Means-end analysis identifies plans, resources, and soft goals that serve as means to achieve a goal, while contribution analysis evaluates how other goals can positively or negatively impact the goal being analysed. AND/OR decompositions subdivide a root goal into sub-goals, creating a more detailed goal structure. Goal modelling is applied to early and late requirement models in order to refine them and to elicit new dependencies. During architectural design, it contributes to motivate the first decomposition of the system actors into a set of sub-actors [15].

Plan Modelling: complements goal modelling and uses the same reasoning techniques: means-end, contribution analysis and AND/OR decomposition [15].

Capability Modelling: It takes place during the latest steps of architectural design, the process involves identifying capabilities for each system actor based on their goals, tasks, and dependencies [11]. Each system's sub-actor needs specific "individual" capabilities to define, select, and execute plans for its own goals, while additional "social" capabilities are required to manage dependencies with other actors [14]. When the agents of the system have been identified, the capabilities corresponding to each of these agents are further specified in detailed design [15].

Secure Reference Diagram modelling: involves the "identification of security needs of the system-to-be, problems related to the security of the system, such as threats and vulnerabilities, and also possible solutions (usually these solutions are identified in terms of a security policy that the organisation might have) to the security problems" [11]. This diagram is constructed after analysing the security requirements of the system-to-be and its environment and represents the relationships between some of the security entities of Secure tropos methodology, namely, security features, threats, protection objectives, and security mechanisms [11]. The primary purpose, as explained by

Mouratidis, is to provide flexibility during the development stages of a multi-agent system, saving time and effort. This diagram serves as a reference point that can be adjusted or expanded to suit the specific needs of individual systems, given the similarity between many systems under development and existing ones.

Security constraint modelling involves representing the security constraints imposed to actors and the system, enabling developers to analyse these constraints and their relationships or security constraints and their context [11]. Activities include security constraint delegation and assignment. Delegation involves transferring a security constraint from one actor to another, while assignment associates a constraint with a specific actor goal. Activities also include analysis to find more detailed security constraints. Indeed, security constraints can introduce goals to actors, known as secure goal introduction, to assist in meeting the constraint. This process involves refining actor's goals to satisfy security requirements.

According to Mouratidis, developers may combine these activities together with other modelling tasks like goal or task transformations to define the system based on imposed security constraints. He further explains that the designers have the flexibility to choose which activities to employ at different stages, as the aim is to provide a flexible framework for transitioning from high-level design to a more precise system definition without strict procedural rules [11].

***Secure entities modelling*:** involves the analysis of the secure entities of the system such as secure goals, tasks and resources. This analysis is considered complementary to the security constraints modelling and uses similar techniques as Tropos, such as means-end analysis, contribution analysis, and AND/OR decomposition. Means-end analysis identifies secure tasks and resources needed to achieve a secure goal. Contribution analysis helps identify secure goals that affect the analysed goal positively or negatively. AND/OR decomposition decompose secure goals and tasks into smaller goals/tasks [11].

Secure entities modelling diagrams are based on existing Tropos modelling activities and they have been extended with respect to security modelling. Extended diagrams include Actor diagram, Goal diagram**,** Plan diagram, Agent interaction diagram…

For example,

Actor diagram => Security enhanced actor diagram

Goal diagram => Security enhanced goal diagram

Secure capability modelling: involves the identification of the secure capabilities of the actors and the agents of the system to guarantee the satisfaction of the security constraints. It takes place together with the capabilities modelling during the architectural design. Secure capabilities can be identified by considering dependencies that involve secure entities in the extended actor diagram [11].

These diagrams mentioned above are created during the different phases of the Tropos methodology, and by extension, Secure Tropos. The distribution of the diagrams according to the phases is as follows:

In the early requirements analysis step, modelling activities include:

- Security enhanced actor diagram;
- Security enhanced goal diagram;
- Security reference diagram

During late requirements analysis, we create:

- Security enhanced actor diagram with the system;
- Security enhanced system goal diagram

Architectural design involve modelling:

- System decomposition diagram;
- Architectural design;
- Security attack testing;
- Security attack scenarios.

Finally in detailed design, the following diagrams will be produced:

- Capabilities diagram;
- Plan diagram;
- Agent interaction diagram.

4. Near-Field Communication

Near Field Communication (NFC) is a technology that enables contactless, short-range communication between two devices, typically within 10 centimeters or less. The technology behind NFC is very similar to Radio Frequency Identification (RFID) technology but is an evolution of it, offering more advanced features and better security. NFC operates using two primary modes: active and passive. In active mode, both devices generate their own radio frequency fields to communicate, whereas in passive mode, only the initiator device generates an RF field, and the target device responds by modulating this field. This modulation allows the transfer of data between the devices. Common applications of NFC include mobile payments, contactless ticketing, and data exchange between devices like smartphones and tablets [16].

5. Non-Fungible Token

A Non-Fungible Token (NFT) is a unique digital asset that uses blockchain technology to verify ownership and authenticity. Unlike cryptocurrencies such as Bitcoin, which are fungible and can be exchanged on a one-to-one basis, NFTs are unique and cannot be exchanged identically. Each NFT is encoded with distinct information, typically representing digital art, music, video clips, or in-game items. The uniqueness and ownership of an NFT are verified through smart contracts on blockchain platforms like Ethereum.

Despite their advantages, NFTs face several security issues. One major concern is the risk of spoofing, where attackers might steal private keys to gain unauthorized access to NFTs. Another risk is data manipulation, particularly for data stored off-chain, which could be altered by malicious actors [17].

6. Raspberry Pi

A raspberry is a series of small single-board computers developed by the Raspberry Pi Foundation to promote computer science education [18]. It's widely used for educational purposes, projects such as home automation, robotics, and media centers, and industrial applications like prototyping and automation. Raspberry Pi runs Linux, but it also provides a set of GPIO (general-purpose input/output) pins, allowing control of electronic components for physical computing and exploration of the Internet of Things (IoT). However, it can face security issues like a lack of built-in security, vulnerability to physical access, and software vulnerabilities that need regular updates.

7. Related Works

In recent years, people have become really interested in blockchain technology due to its potential to change various industries. In fact, blockchain has fundamentally changed how we ensure data integrity and manage storage, offering unequalled levels of transparency, security, and decentralization. This technology has improved sectors such as finance, supply chain management, and healthcare by enabling secure, tamper-proof transactions and decentralized database system. However, as the development of blockchain projects progressed, challenges emerged, such as security vulnerabilities in smart contracts and the failure of some blockchain projects initiatives. So, researchers started looking into ways to model these blockchain systems better. Here we go through a number of available researches in this area to better understand how people are trying to make blockchain modelling work better.

Paper [19] presents the extension of i* framework (a modelling and reasoning framework relevant to model organizational requirements of multi agent system) with privacy and legal compliance concepts to address challenges related to privacy, data integrity, and regulatory compliance in blockchain adoption for SCM. Authors provide strategic dependency and rationale diagrams using i* to illustrate the interdependencies between various stakeholders and the blockchain system. Furthermore, they use UML diagrams like Use Case and Sequence diagrams to depict system interactions and data flow. They similarly compare their strengths and weaknesses and acknowledge limitations in both approaches to model BOSE in SCM to its full extent. As a future work, they suggest enhancement to address gaps and the development of transformation rules to support the blockchain implementation with object and agent technology.

Article [20] introduces a new approach named Blockchain-Oriented Service Modelling, specifically designed for developing blockchain-enabled Supply Chain Quality Information Systems (SCQIS). The authors' objective is to model services by capturing the interactions and collaborative value generation among supply chain partners. They capture both the service management and service computing perspectives and separate operant resources (the root cause of quality) from products (the carriers of quality) to enhance defect detection and inspection processes.

Additionally, authors claim that service-oriented modelling framework facilitates the coordination and collaboration among stakeholders by providing a structured representation of their interactions and dependencies. However, it's worth pointing out the fact that service modelling does not allow to verify the reliability of service and the reasoning behind it as service is taken as a block box.

Article [21] extends the Secure Tropos methodology by including GDPR and privacy concepts. The author also introduces a new modelling language, Privacy Enhanced Secure Tropos (PESTOS) (meta-model, semantics, and concrete syntax), which is a “privacy modelling language based on Tropos methodology, which covers the goal and rule perspective, to help software engineers by assessing candidate privacy-enhancing technologies (PETs), while designing privacy-aware systems in order to make them compatible with GDPR” [21].

In the same paper [21], the author shows that “Secure Tropos language does not provide support for modelling legal dependencies which are needed for eliciting privacy requirements from legal contexts such as GDPR text”.

Paper [7] examines existing MDE techniques for BBAs, emphasizing the need for secure and efficient development methods to address the criticality of errors, particularly in smart contracts. Authors propose MDE as a solution to overcome these challenges and therefore facilitate the development of BBAs. However, the study reveals several gaps in current MDE techniques, including the lack of support for data privacy and formal verification. Additionally, certain elements, such as interaction with third-party smart contracts and representation of assets on the blockchain, are inadequately modelled, while automatic code generation for blockchain oracles is not widely supported.

Article [22], emphasizes the challenges and specific requirements involved in modelling BOSE, particularly focusing on data aspect. Authors highlight the current lack of modelling techniques tailored specifically for blockchain applications and propose an extension to the Entity-Relationship (ER) model to fill this gap. They introduce stereotypes for smart contracts and other blockchain entities like, structs, variables, mappings, addresses, and ledgers, allowing for clear differentiation between blockchain and traditional database entities, and also providing clarity and precision in modelling BOS. However, there still have limitations in fully capturing the complexities of blockchain data modelling. Their model only works for one blockchain network and doesn't handle multiple networks.

The related studies either suggest new ways to model BBS or give an overview of their current modelling situation. But they all agree on a common conclusion: there are significant gaps in how we model these systems. These gaps can be noticed in various aspects, such as the structure of the proposed models, their legal compliance with regulations like GDPR, or their security. So, it's clear that we need better ways to model blockchain systems to solve these problems and make blockchain technology much more useful.

III. RESEARCH METHODOLOGY

This section will lay the focus on the research methodology while highlighting such key aspects as the selected methodology, research questions and research process.

1. Choice of Methodology

This paper proposes a specifically tailored methodology for Blockchain based software development. But prior to addressing this issue, it is essential to clearly understand blockchain systems, for example how they're modelled and what potential limitations they are likely to face. That is the reason why I conducted a Systematic Literature Review (SLR), the most suitable methodology to gather valuable insights in my opinion.
Furthermore, blockchain comprehensive modelling still remains largely unexplored and existing researches do not often pay enough attention to security, thus implying a clear need for relevant strategies and methods to address these gaps.

Owing to its effective security modelling, I personally prefer to extend Secure Tropos. In addition, since no information was available throughout my investigations, so far, on any real attempt at combining Secure Tropos with blockchain before, I felt that opting for a Proof of Concept (POC) would make sense to me.

It allows me to propose a new solution and show that it works in real life situation. Through POC, I can demonstrate how the new method works and therefore show how it can meet expected outcomes, thus encouraging its use and even its gradual improvement within the research community.

2. Research Questions

This master thesis aims at providing an easy-to-use and customized methodology for Blockchain based software development. This methodology will extend the agent oriented « Secure Tropos » system to address BBS and related security aspects, therefore allowing developers to better implement these systems while keeping security concerns in mind at the early stage of the development process. To meet this concern, my main research question (MRQ) is: "How to extend the Secure Tropos methodology to build Blockchain-based software systems?".

My MRQ is further split into three (3) sub-questions to guide the investigation process.

1. RQ1- What aspects of BBS should be considered during its development? This question will explore the various aspects of BBS that should be taken into consideration during its development regarding BOSE principles.

2. RQ2- How can Secure Tropos methodology implement Blockchain principles? This question seeks to understand the integration of Blockchain principles within the Secure Tropos methodology in an effective and valuable way.

3. RQ3- How to apply enhanced Secure Tropos methodology to BBS development? This question investigates the practical application methods and steps of the proposed methodology in BBS development.

3. Research Process

This research lies on two (2) distinct approaches (Figure 3), each one representing a gradual step to meet the final outcome. The research steps are:

First step: the State-of-the-Art phase

I adopted a state-of-the-art methodology to address RQ1 and RQ2 in my research. I conducted a SLR to get a deeper understanding of BBS. Through the SLR, I identified and analysed the essential aspects in developing BBS, including modelling techniques and their limitations. This process allows me to examine blockchain systems from both structural and security perspectives in order to point out valuable assets for the new methodology. The SLR findings will serve as a sound basis for aligning the secure tropos methodology with blockchain while enriching it with new concepts relevant to blockchain technology.

Kitchenham [23] has been used for the first step because it is one of the best SLR guidelines adapted to software engineering research guidelines. The authors present three (3) phases of a SLR: planning, conducting, and reporting.

1. **Planning**: prior to the review and including the following activities: justifying the review, setting specific research question(s), and launching a review protocol.
   Regarding this SLR, these steps have been monitored to confirm the research ability and feasibility on the review, define the research question and sub-questions, as well as the inclusion and exclusion criteria.
2. **Conducting**: the review activity begins here. Once my research protocol set up, I gradually and iteratively went through: research identification, primary studies selection, quality assessment study, and, finally data extraction, monitoring & synthesis. This helps me answer my questions.
3. **Reporting**: the review outcome, involving review results drafting and documenting via: information sharing mechanisms, report drafting and evaluation. In my own situation, the present paper (examined by my supervisor) is the result of this phase.

I used Scopus and Google Scholar to search for scientific papers.

Next step: the POC

This paper is also following the POC guidelines from Cambridge university press [24]. In that guideline, three (3) phases of a POC are being covered by the authors: prototypes, POC demonstrations, and post facto arguments.

1. The prototype phase (referred to as “proposal of solution” in my paper) will be dedicated to building the Blockchain-oriented Extension of Secure Tropos by using the ontology, and knowledge reasoning from step 1.
2. The proof-of-concept demonstrations phase will allow me to go through an effective assessment of proposed secure tropos methodology extension to clearly establish that the prototype is achieving expected results. The validation will be done through a case study on the Smart

Blockchain Supply-chain project (BC24-Trace) done by MIAGE students at University Paris 1 Pantheon Sorbonne.

3. The post facto arguments phase will be a discussion on the results and will use heuristics to argue pragmatically for the further development of prototype as recommended by Elliot in [24].

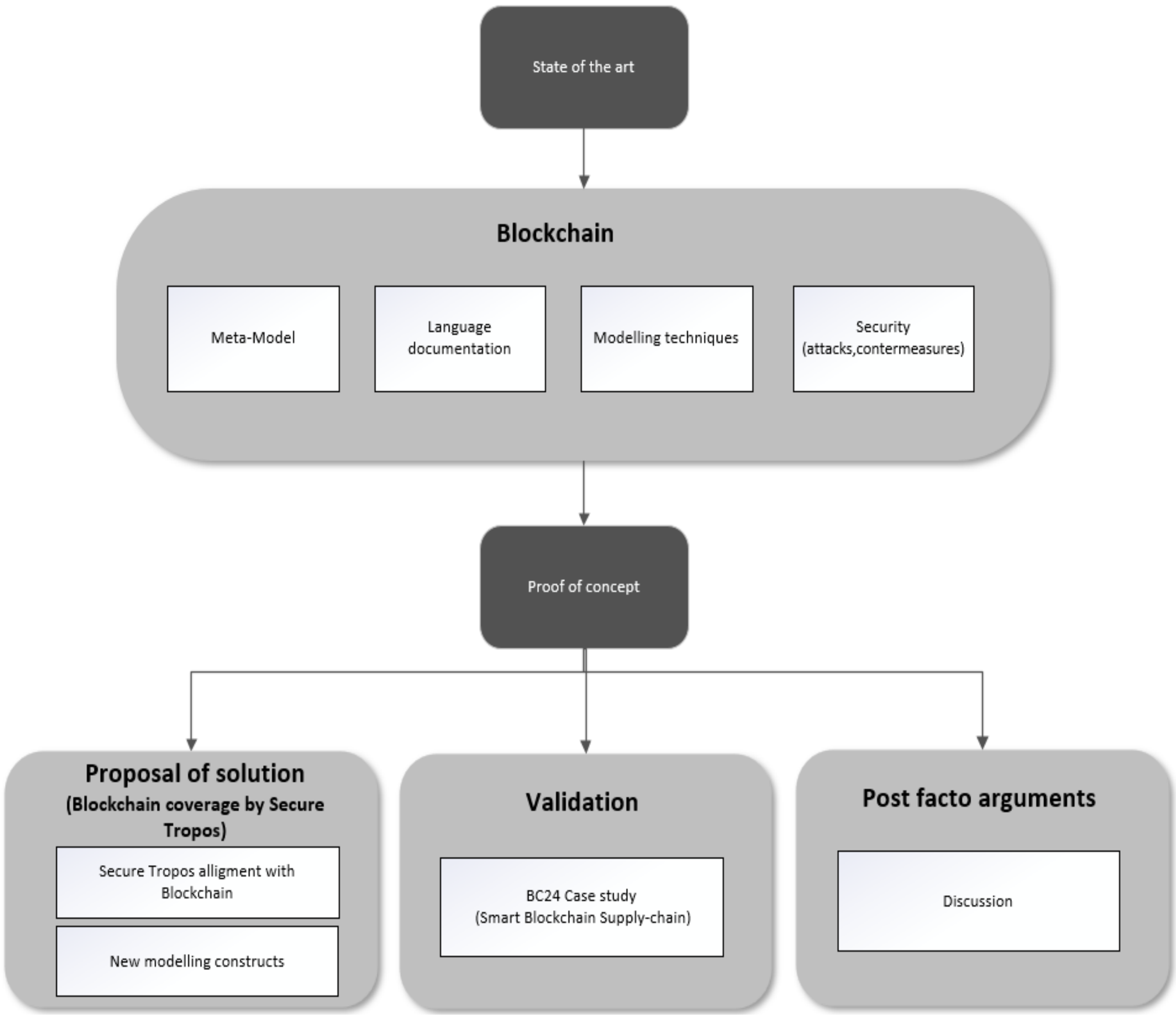

State of the art
Blockchain
Meta-Model
Language documentation
Modelling techniques
Security (attacks,contermeasures)
Proof of concept
Proposal of solution (Blockchain coverage by Secure Tropos)
Secure Tropos alligment with Blockchain
New modelling constructs
Validation
BC24 Case study (Smart Blockchain Supply-chain)
Post facto arguments
Discussion

Figure 3: Research methodology.

IV. SECURE TROPOS FOR SECURITY & BOSE MODELLING

In this section, the Secure Tropos concepts will be examined to determine how blockchain concepts are overlooked and similarly explain the reason supporting the idea of extending the Tropos methodology.

1. Overlooked Blockchain in Secure Tropos

The examination of the Secure Tropos concepts reveals the following points concerning overlooked blockchain aspects:
In Secure Tropos, a security constraint is defined as a restriction related to security issues such as privacy, integrity, and availability. These constraints as described by Mouratidis [11], can influence the system's analysis and design by limiting design options, conflicting with system requirements, or refining system objectives. However, while Secure Tropos addresses these traditional security issues, it overlooks blockchain-specific security attributes such as partition tolerance and non-repudiation, etc. These attributes, among others, are at the core of the proper functioning and security of blockchain systems. Not being sufficiently covered by Secure Tropos, they must be added to avoid incomplete or wrong security designs for blockchain environments.

Moreover, Secure Tropos added security notions (secure goals, secure tasks, and secure resources) to the Tropos entities to enhance the ability to model more systems and their security. However, even with these improvements, the methodology is still not ready to adequately model some blockchain components and resources, like smart contracts and transactions. For a blockchain designer, it is simply inconceivable to create a blockchain system without being able to model transactions, its core unit. These elements are non-negotiable and must be explicitly modelled. Therefore, it is necessary to add BC entities such as BC goal, BC task, and BC resource to Secure Tropos to accurately reflect in the model, their functions within a blockchain network.

Furthermore, Secure Tropos provides the possibility to test the system under development against potential attacks by using security attack scenarios. Mouratidis [25] defines a Security Attack Scenario as an attack situation involving the actors in a software system, their security measures, and potential attackers with their goals. It shows how the system's security measures can (or cannot) prevent the attackers from achieving their objectives. This testing activity takes place at the end of the architectural design, where the design of the system is tested against the security requirements. This addition is particularly interesting because it can be applied to any system, including a blockchain system. However, to be more comprehensive, we should include smart contract testing. Although this can be included within the framework of security attack scenarios, it is better to give it special attention because it can be a challenge for blockchain systems. In fact, smart contract testing is crucial for several reasons. First, no change is possible on a deployed smart contract due to the immutable nature of blockchain. Second, smart contracts often handle valuable assets and sensitive information. Any vulnerability can be exploited, leading to significant financial losses or data breaches. Therefore, it is important for smart contract developers to fully test and verify the correctness of their code before deploying it on the blockchain.

2. Motivations for Secure Tropos Extension (Why it is needed?)

Several reasons can justify why extending Secure Tropos with a Blockchain-oriented approach is necessary.

First, as we mentioned before, security is a critical issue for any system, especially those using blockchain characterized by its decentralization, thus enforcing the need to set forth strong security measures. Similarly, most parts are interdependent and interrelated in a blockchain system; so, each part will need to make sure the others are properly achieving their roles and meeting expected outcomes.

Secondly, it's been noticed that Secure Tropos language does not cover legal dependencies necessary for eliciting security needs from a legal perspective, as highlighted by Islam et al. in their study [26]. As a result, key elements likely to help determine what security features are necessary (based on legal requirements) are missing, potentially leading to vulnerabilities in the system. These critical constraints might be very expensive to fix and cope with later if found after the application is launched. Much money and manpower could be required to overcome these constraints internally, or severe penalties may be faced adversely from regulators like GDPR in case of observation.

Furthermore, as shown in the section, the existing notations within the Secure Tropos framework do not adequately support blockchain components or represent blockchain-related concepts.

Therefore, it's important to extend this method to better model them and, ultimately build them as strong systems. This master thesis will extend Secure Tropos in order to support blockchain and blockchain security considerations during the development of sensitive software blockchain systems. The solution will enable software developers to:

- (1) correctly elicit security requirements for blockchain system;
- (2) correctly elicit smart contract development requirements for blockchain system;
- (3) correctly elicit requirements from norms and regulations applied to the blockchain system to build;
- (4) trace these requirements during the development stages to fully ensure design phase can cope with building compliant systems;
- (5) consider security, privacy and blockchain issues simultaneously with the system-to-be other requirements.

The proposed system will extend the Secure Tropos methodology by introducing new blockchain-related concepts and processes, including blockchain-oriented processes, smart contract-oriented development processes while integrating these processes into the Tropos methodology gradual development levels.

Elements	Secure Tropos	Proposed extension
Actor	Yes	Yes
Intentional elements (goal, task, resource, capability)	Yes	Yes
Specific blockchain entities (partition tolerance, non-repudiation)	Partially (with security features)	Yes
Privacy considerations (A, C, PA, AC)	Partially (with security constraints)	Yes
Legal dependencies	No	Yes
Conflicting legal dependencies	No	Yes
Specific blockchain development requirements (smart contract testing)	No	Yes
Specialized blockchain requirements (patterns, anti-patterns…)	No	No

Table 1: Comparison between Secure Tropos and the proposed methodology regarding their implementation level of blockchain concepts.

The table 1, summarizes Secure Tropos' gap analysis regarding BBS modelling. It compares the capabilities and limitations of Secure Tropos with those of our proposed methodology. "No" indicates that the methodology does not support the corresponding aspect, whereas "Yes" means it does support this aspect.

V. BLOCKCHAIN-ORIENTED EXTENSION OF SECURE TROPOS

This section will focus on presenting the extensions made to Secure Tropos to support the design of BBS. It covers the semantics, concrete syntax, new constructs, and the application method. Figures will be included throughout this section to summarize key points and present new constructs.

1. Semantics and Syntax

The entities of the Tropos methodology need to be extended with the principles of BOSE. This begins by aligning the concepts of Secure Tropos with BOSE before introducing new constructs.

i. BOSE alignment with Secure Tropos

Blockchain (BC) Actor:

An actor represents an entity that has intentionality and strategic goals within the multiagent system or within its organisational setting. An actor can be a (social) agent, a position, or a role. Agents can be physical agents, such as a person, or software agents [13].
In Blockchain, there are several actors (agents) and components having defined goals, that play a major role to make the network more secure and reliable. Those Blockchain actors are:

- ***Ledger:*** A ledger is a current and previous state data storage maintained by each node on the network [27].
- ***Blockchain or Blockchain Network***: A blockchain network is constituted of peers, each owning a copy of the ledger and listening to each other to keep their version up to date.
- ***Wallet:*** is the software or app where people keep their Bitcoins. Wallets include desktop wallets, mobile wallets, and hardware wallets. Wallets allow consumers pay Bitcoin for their purchases. Thus, they play an important role in wider adoption of Bitcoin.
- ***Node:*** refers section “Blockchain’s components”.
- ***Smart contracts***: refers section “Blockchain’s components”.

Concepts from Blockchain	Concept from Secure Tropos	Blockchain constructs
Ledger	Actor	○ BC Actor
Blockchain or BC network	Actor	○ BC Actor
Wallet	Actor	○ BC Actor
Node	Actor	○ BC Actor
Full node	Actor	○ BC Actor
Lightweight node	Actor	○ BC Actor
Smart contract	Actor	○ BC Actor

Table 2: Blockchain actors.

In this table 2 we present some examples of blockchain actors modelled with Secure Tropos existing constructs. As we can see, the same notation of actor (circle) has been used because the actor concept in Secure Tropos allows modelling any type of actor. There are no particular differences that need to a specific notation, in fact a blockchain actor has strategic goals and beliefs within its organisational setting like a normal actor do. But to specify that it is a blockchain actor, one can add (BC) label, it is not mandatory it is just for more convenience. We recommend adding the label to let understand that we are in the context of blockchain modelling. Sometimes, an element can exist in another context and have a slight difference in meaning when it comes to blockchain, therefore it is important to set the context with the label. This procedure can be used for all other secure tropos concepts that can be translated for blockchain modelling.

BC Resource: a physical or informational entity that is blockchain-critical and which the blockchain actor requires in order to perform a task. An example of BC resource is a transaction. The BC resource can be graphically represented as a Resource or as a Resource with a (BC) label additionally. Both notations can be used but we recommend adding the label for the reasons mentioned in the description of table 2.

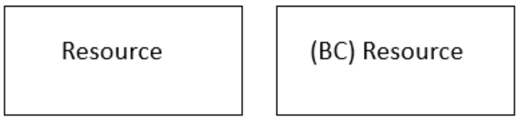
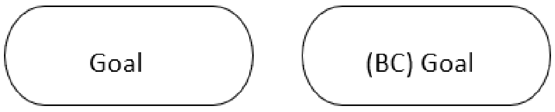


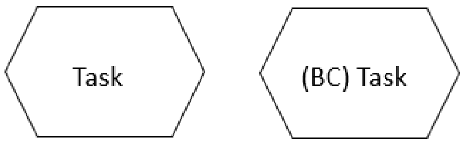
Figure 4: Possible graphical representation of BC resource.

<u>BC Goal</u>: represents the strategic interests of a blockchain actor with respect to BOSE defined best practices and aspects in the design and modelling of blockchain systems. BC goals may concern consensus activities like validation, node election… For instance, an example of BC goal is "the validators must check the validity of the previous block referenced by the current block". The BC goal can be graphically represented as a Goal or as a Goal with a (BC) label additionally. Both notations can be used but we recommend adding the label for the reasons mentioned in the description of table 2.



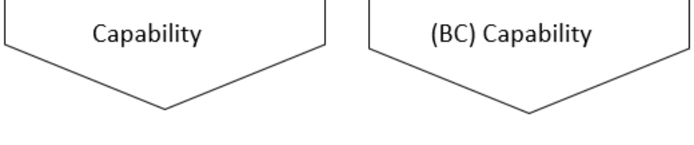
<u>Figure 5:</u> Possible graphical representation of a BC goal.

<u>BC Task:</u> represents a way of satisfying a BC goal. More simply, it represents actions that a BC actor wants to perform in order to satisfy its BC goals. Task is also called plan in Tropos. The BC task can be graphically represented as a Task or as a Task with a (BC) label additionally. Both notations can be used but we recommend adding the label for the reasons mentioned in the description of table 2.



<u>Figure 6</u>: Possible graphical representation of a BC task.

<u>BC Capability</u>: represents the ability of an BC actor to define, choose, and execute a task for the fulfilment of its BC goal, given certain world conditions and in presence of a specific event. Its graphical representation (shown below) following the same technique of introducing an (BC) within brackets before the capability label, to depict Blockchain capabilities. Note that it could also be represented as a Capability.



<u>Figure 7</u>: Possible graphical representation of a BC capability.

Concepts from Blockchain	Concept from Secure Tropos	Blockchain constructs
Transaction	Resource	(BC) BC Resource
Consensus	Task	(BC) BC Task
Block validation	Goal	(BC) BC Goal

Table 3 : Secure Tropos and Blockchain aligned concepts.

In this table 3, we present some examples of blockchain intentional elements modelled with Secure Tropos existing constructs. Notations are the same as in Secure Tropos. But we recommend adding a (BC) label to better reflect their function within a blockchain network like it is mentioned in the description of table 2.

Blockchain Security reference diagram:

The construction of the security reference diagram involves considering these elements: the security features of the system-to-be, the protection objectives, the security mechanisms, and the threats to the system's security features. These same elements are used for the blockchain security reference diagram, with a slight difference in the security features.

Blockchain security Features*:* According to Mouratidis, security feature are security aspects like privacy, integrity, and availability that a system must have. From BOSE's perspective, there is a need to redefine "Blockchain Security Features". In addition to existing features, this new definition will consider the security attributes of blockchain such as consistency, partition tolerance, etc.

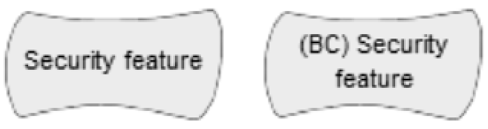
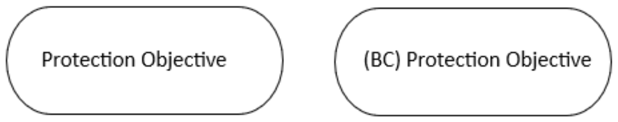


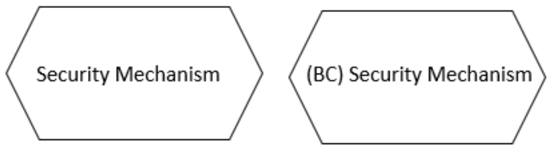
Figure 8: Graphical representation of a security feature and a BC security feature.

Blockchain Protection Objectives**:** It is the same as proposed in Secure Tropos. They represent a set of principles or rules that contribute towards the achievement of the security features. These principles identify possible solutions to the security problems and usually they can be found in the form of the security policy of the organization [11]. The BC Protection Objectives can be graphically represented as a Protection Objectives or as a Protection Objectives with a (BC) label additionally. Both notations can be used.



<u>Figure 9</u> : Possible graphical representation of a BC protection objective.

Blockchain Security Mechanism**:** It is the same as proposed in Secure Tropos. It represents standard security methods for helping towards the satisfaction of the protection objectives. Mouratidis claims that some of these methods are able to prevent security attacks, whereas others are able merely to pinpoint security breaches. The concept of a task is used to model security mechanisms in Mouratidis' work. As he clarifies, this decision took place, because in Tropos, a task represents a particular way of doing something, such as the satisfaction of a goal. In the same sense, a security mechanism represents a particular way of satisfying a protection objective [11]. Its graphical representations are represented below.



<u>Figure 10</u>: Possible graphical representation of a BC security mechanism.

Threats**:** They represent circumstances that have the potential to cause loss; or problems that can put in danger the security features of the system [11].

Now, let's move on to the new Secure Tropos concepts.

ii. Introducing 'Norm and laws' and 'Decision link' concepts

As blockchain evolves and people interests are growing, there's a possibility of new standards and regulations being introduced that could restrict the functionality of blockchain and/or smart contracts [19]. For example, ISO, the International Organization for Standardization is working to establish standards for blockchain in domains like information security, data governance, and system interoperability. Therefore, addressing standards and regulations in blockchain and smart contracts modelling is important for ensuring legal compliance and trust among users. Efforts have been made to expand the Secure Tropos framework to fit the GDPR regulation and privacy concerns [21], but it's not enough. It only focuses on the GDPR and doesn't cover all the other laws and norms that could affect blockchain projects as blockchain is decentralized. So, including a general concept of laws and norms in the Secure Tropos framework is a critical challenge.

To achieve this, I use the concept of Norms from [19, 28, 29] which was introduced to i* framework to model laws and norms.

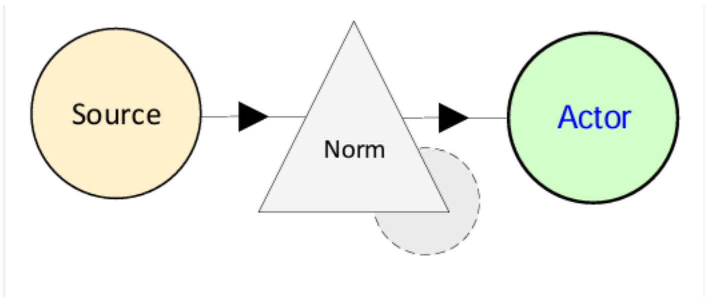


Figure 11: Norm and laws concept, from [19]

Additionally, I have introduced the concept of "Decision link". This is link between two processes with the same goal but different, conflicting tasks. In fact, in the context of blockchain, it is possible to encounter situations where two different countries have laws with the same goal but different methods of achieving it. And tasks required by these laws may conflict or be incompatible. Therefore, system managers need to decide on how to mitigate the issue and which tasks to prioritize. This is where decision link will help by allowing designers to visually identify potential problems early in the design phase and plan their resolution. It is represented by a double arrow to highlight critical points to bear in mind.

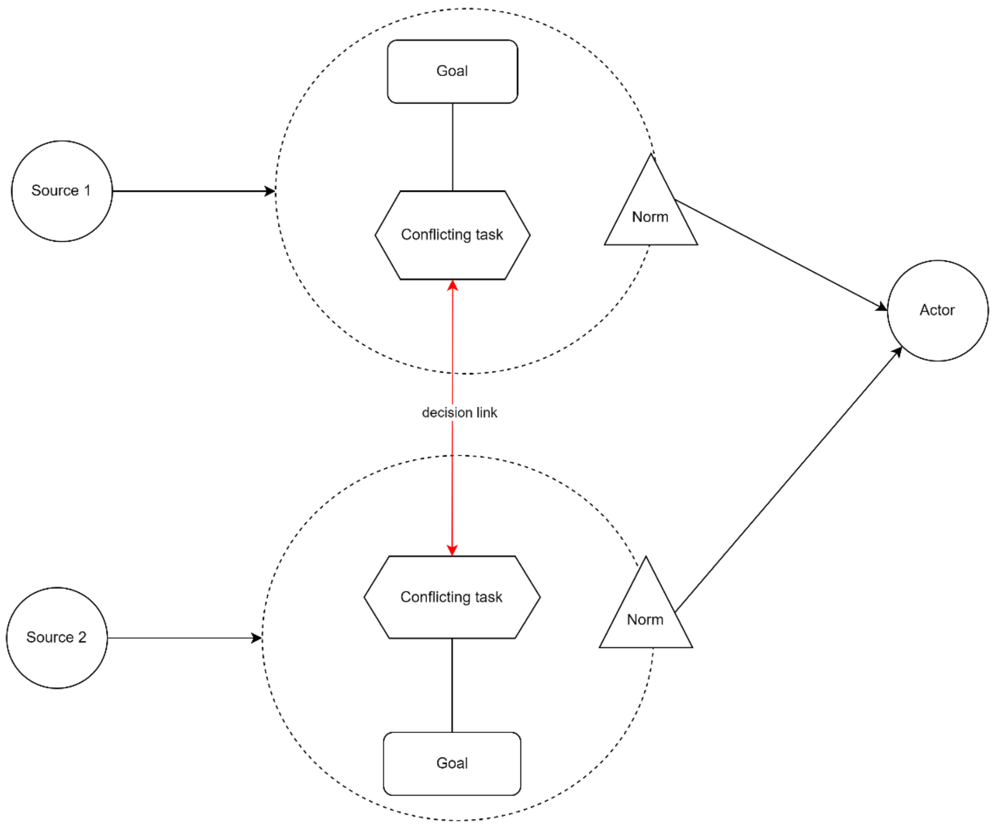


Figure 12: Decision link representation.

iii. Introducing ‘Privacy’ concept

One of the most difficult problems with blockchain technology is privacy. Sometimes, nodes need to decide which data should be anonymized and/or restrict access to some data. Therefore, it is important to explicitly model privacy constraints when designing BBS.

Four privacy concepts were added to i* framework by Ben Hamadi et al [30] and the authors of [19] implemented them in BOSE. These are:

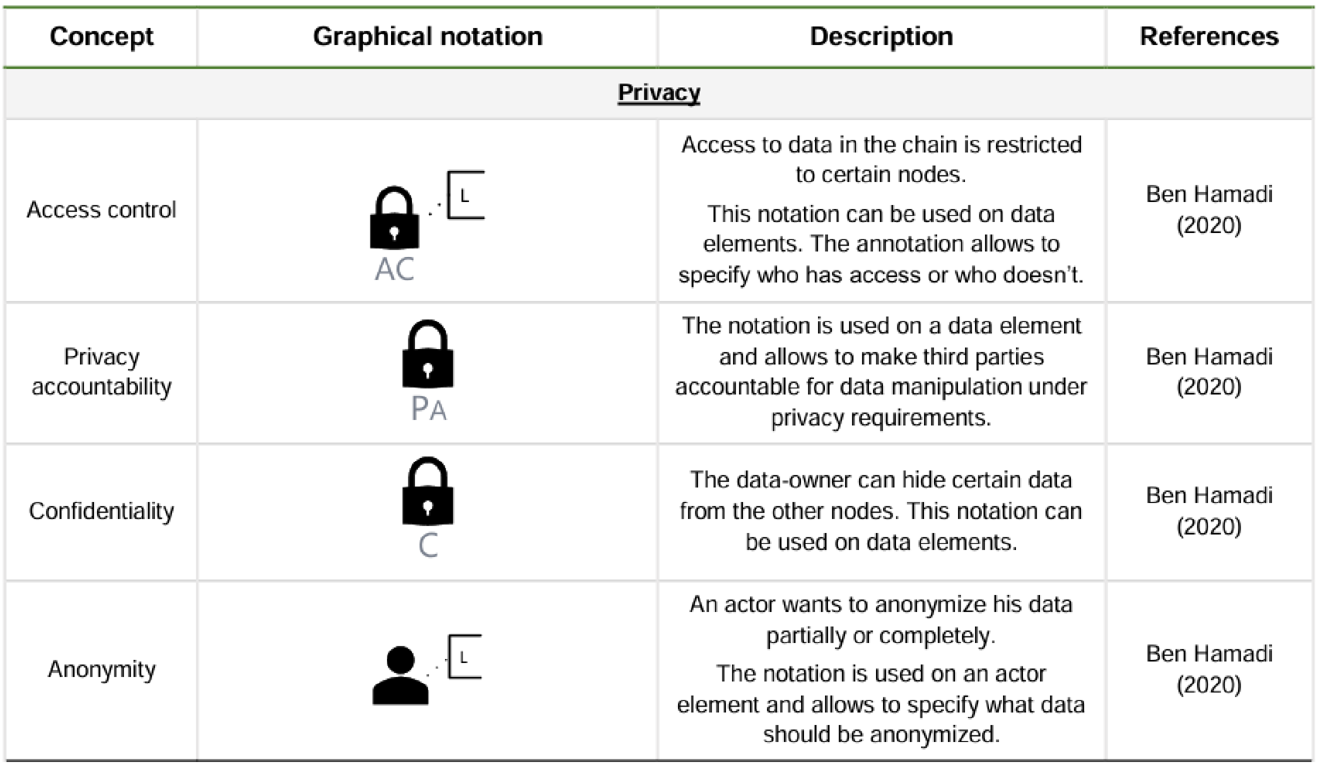
1. Access Control (AC): restricts access to data within the blockchain to certain authorized nodes.

2. Privacy Accountability (PA): ensures that third parties are accountable for data manipulation under privacy requirements.

3. Confidentiality (C): allows data owners to hide some data from other nodes.

4. Anonymity (A): enables actors to anonymize their data partially or completely.

Since these concepts align with my goals to extend Secure Tropos to support blockchain based systems, I have decided to integrate them into Secure Tropos. The elements and their notations as proposed by the authors of [19] are detailed in Figure 13.



Concept	Graphical notation	Description	References
Privacy			
Access control	🔒 L AC	Access to data in the chain is restricted to certain nodes. This notation can be used on data elements. The annotation allows to specify who has access or who doesn't.	Ben Hamadi (2020)
Privacy accountability	🔒 PA	The notation is used on a data element and allows to make third parties accountable for data manipulation under privacy requirements.	Ben Hamadi (2020)
Confidentiality	🔒 C	The data-owner can hide certain data from the other nodes. This notation can be used on data elements.	Ben Hamadi (2020)
Anonymity	👤 L	An actor wants to anonymize his data partially or completely. The notation is used on an actor element and allows to specify what data should be anonymized.	Ben Hamadi (2020)

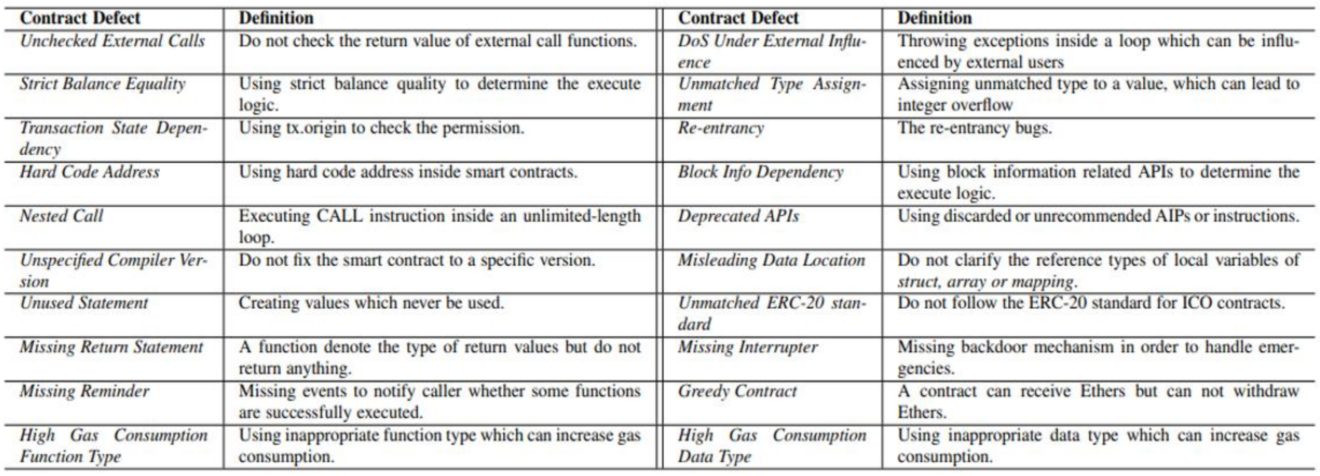
Figure 13: Privacy concepts, from [19]

Privacy is certainly important, but it is not more critical than the security and reliability of smart contracts, which can lead to very chaotic situations if deployed with vulnerabilities. Therefore, it is essential to integrate smart contract testing, which is the focus of our next proposal.

iv. Introducing ‘Testing Criteria’ concept

The idea is to integrate a defect taxonomy into Secure Tropos for blockchain applications. It focuses on validating the behaviour and security of smart contracts before their deployment and is a way of ensuring the traceability of smart contract testing.

This integration of smart contract testing into secure tropos begins with a taxonomy of common smart contract defects taken from the literature. This taxonomy classifies defects into three types: security vulnerabilities, performance issues, and compliance gaps [31]. Currently, a set of 20 defects is presented in the Figure 14. These defects are focused on Ethereum, with some general smart contract defects, but this list can be changed in the future to include more general smart contract defects.



Contract Defect	Definition	Contract Defect	Definition
Unchecked External Calls	Do not check the return value of external call functions.	*DoS Under External Influence*	Throwing exceptions inside a loop which can be influenced by external users
Strict Balance Equality	Using strict balance quality to determine the execute logic.	*Unmatched Type Assignment*	Assigning unmatched type to a value, which can lead to integer overflow
Transaction State Dependency	Using tx.origin to check the permission.	*Re-entrancy*	The re-entrancy bugs.
Hard Code Address	Using hard code address inside smart contracts.	*Block Info Dependency*	Using block information related APIs to determine the execute logic.
Nested Call	Executing CALL instruction inside an unlimited-length loop.	*Deprecated APIs*	Using discarded or unrecommended AIPs or instructions.
Unspecified Compiler Version	Do not fix the smart contract to a specific version.	*Misleading Data Location*	Do not clarify the reference types of local variables of *struct, array or mapping.*
Unused Statement	Creating values which never be used.	*Unmatched ERC-20 standard*	Do not follow the ERC-20 standard for ICO contracts.
Missing Return Statement	A function denote the type of return values but do not return anything.	*Missing Interrupter*	Missing backdoor mechanism in order to handle emergencies.
Missing Reminder	Missing events to notify caller whether some functions are successfully executed.	*Greedy Contract*	A contract can receive Ethers but can not withdraw Ethers.
High Gas Consumption Function Type	Using inappropriate function type which can increase gas consumption.	*High Gas Consumption Data Type*	Using inappropriate data type which can increase gas consumption.

Figure 14: Definitions of the 20 contract defects, from [31].

Next, I've introduced a new entity called "testing criteria” (see Figure 15) representing smart contracts development requirements such as security, and certain (number of modifiers, total lines of code to a specific blockchain address...). These entities are linked to the taxonomy of defects, creating a relationship between identified vulnerabilities and the tests designed to address them. For example, if a known defect involves the risk of re-entrancy attacks, the model includes specific test criteria that have to be tested in order to prevent this vulnerability.

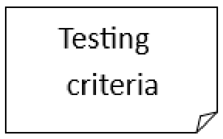
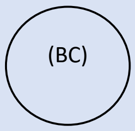
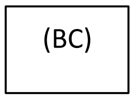
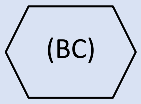
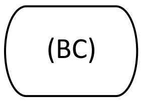
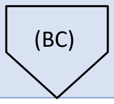
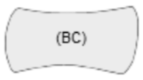
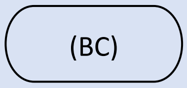
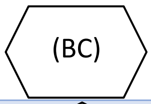
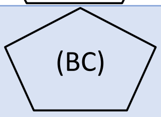
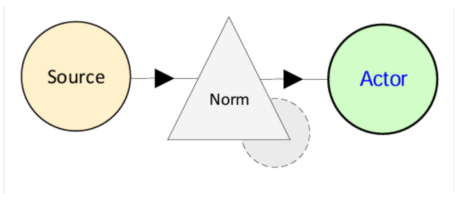
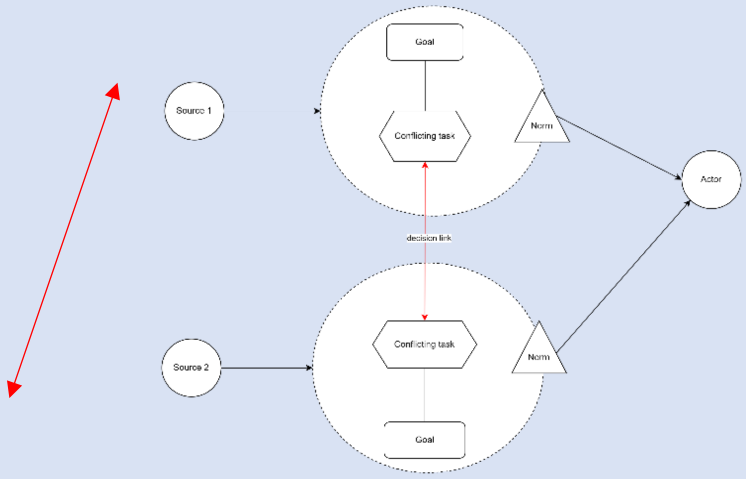


Figure 15: Graphical representation of Testing Criteria concept.

The concepts presented in this section and their notations are summarized in the following table below.

Concept from Secure Tropos	New Blockchain constructs
Actor	 BC Actor
Resource	 BC Resource
Task	 BC Task
Goal	 BC Goal
Capability	 BC Capability
Security Feature	 BC Security Feature
Protection Objective	 BC Protection Objective
Security Mechanism	 BC Security Mechanism
Threat	 BC Threat
N/A (Not Available)	 Laws and Norms
N/A	 Goal Source 1 Conflicting task Norm Actor decision link Source 2 Conflicting task Norm Goal Decision link

N/A	🔒 AC	Access Control
N/A	🔒 PA	Privacy Accountability
N/A	🔒 C	Confidentiality
N/A	👤	Anonymity
N/A	(BC)	Testing criteria

Table 4: Blockchain related concepts in Secure tropos.

2. Application Method

The application method remains the same as the Secure Tropos methodology. There is no need to change it as it performs well, and I do not want to change the habits of secure Tropos users. Instead, I will add activities at each stage to incorporate our enhancements.

During the early requirements analysis, we will:

- Include blockchain constructs retrieved from matching with Secure Tropos;
- Include privacy concepts in the actor analysis;
- Include norms and laws into the actor analysis.

In the late Requirements Analysis, we will:

- Add the smart contract actor when introducing the system actor;
- Apply the testing criteria concept to the added smart contract actor;
- Conduct a deeper analysis of the “norms” elements added during early requirements analysis and identify potential decision links that could be integrated.

For the architectural design and detailed design stages, there will be additions linked to the previous two steps, but the process itself remains the same as in secure Tropos.

The step to follow is presented in the next section “VALIDATION- CASE STUDY”.

VI. VALIDATION- CASE STUDY

The BC24/Trace [32] project aims to develop a POC for a secure manufacturing traceability system. Specifically, it involves implementing a blockchain solution designed to track farm animals throughout the entire supply chain process, from their birth to the point of consumption. The solution will also include the transfer of ownership between different actors in the system.

To achieve this goal, NFC tags will be used to identify various resources (animals, meat carcasses, or meat spare parts), Raspberry Pi devices for tag scanning and reading, NFTs linked to NFC tags, and a blockchain platform where all our NFTs will be stored.

This integrated approach will allow to have a complete tracking system for resources and artifacts. Participants, especially consumers, will have the ability to easily check the origin of the meat they intend to purchase, as well as its health status, quality, and the various stages the animal has undergone, all through a dedicated consumer application.

1. Overview of "BC24-Trace" Blockchain supply chain participants and process

In the "Trace" Blockchain supply chain, various players work closely together, each with their specific tasks and responsibilities. Thanks to its traceability properties, this project has applications beyond the food industry. For this case study, the major focus will be laid on the beef meat's journey, from the farm to the consumer.

The following explanations are summarized in Table 5.

It all starts with the farmer, who raises the beef. When a new animal is born (or arrives at the farm), the farmer sticks an NFC tag on it and scans it to create a digital record in the blockchain. This record includes details like date and place of birth, gender, weight, health information, and food consumption information. Then, a transporter picks up the animals and takes them to the butcher. The transporter scans each animal, marking a change of ownership, and records the duration of the journey.

At the butcher's, the meat is prepared, that is to say the beef slaughtered, cut into carcasses and then pieces (ribeye, ribs, steak…). Slaughter and cutting information like number of slaughtered beefs, cutting country, dates… are logged. Information about factory cutting approval number, slaughterhouse approval number are already in the system, and others like slaughter country are taken form GPS.

Once the animals become meat, another transporter takes the carcasses to the factory. They scan each one, take ownership and note conditions like temperature, humidity, and transport duration. At the factory, workers take ownership and create recipes and package the meat for supermarkets. Using a graphical interface and touchscreen buttons connected to the Raspberry Pi, the user can select a recipe (merguez, steak…). Depending on the chosen recipe, the user may need to select multiple meat cuts. In this case, the resulting digital records will combine different meat cuts NFT and additional information like Date of manufacturing, Product Name, Description, Weight, and references to all used ingredients. Next comes another transporter that takes ownership and moves the meat and manufactured products to the supermarket, scanning each item along the way. At the Supermarket, the merchant scans and takes ownership of the products, then registered prices for products. Finally, consumers can buy the products and check their origin and quality using a dedicated app.

Blockchain Participants	Tasks	Information stored in blockchain
Farmer	Raises the animals (beef)	Breeding information: Beef characteristics: • Place of birth (HE) • Date of birth (HE) • Gender (NHE) • Weight (NHE) Health information • Sickness (NHE) • Date of sickness (System) • Vaccine (NHE) • Date of vaccine (System) Food consumption • Food type (NHE) • Food-quantity (NHE) • Date of consumption (System)
Transporter	Transports the beef from the farm to the butcher.	Transportation information Duration (HE)
Butcher	Prepares beef: Slaughters animals cuts into meat carcasses. Cuts meat carcasses into pieces (ribeye, ribs, steak…)	Slaughter information: • Slaughter country (HE) • Slaughterhouse approval number (System) • Slaughter date (System) • Number of beefs slaughtered • Treatment conditions Cutting information • Factory cutting approval number (System) • Cutting date (System) • Carcass chosen (System) • Quantity of pieces (NHE) • Cutting country (HE)
Frozen Transporter	Transports the meat carcasses from the butcher to the factory.	Transportation condition • Humidity levels (HE) • Temperature conditions (HE) Transportation information • Duration (HE)
Factory	Prepares meat recipes (merguez, steak…) Packs the meat pieces and recipes according to needs of the supermarket.	Factory information • Factory name • Factory country Product information

		• Date of manufacturing • Product Name • Description • Weight • Carcass chosen • Meat amount • Ingredients
Frozen Transporter	Transports the meat and meat recipes from the factory to the supermarket.	Transportation condition • Humidity levels (HE) • Temperature conditions (HE) Transportation information • Duration (HE)
Merchant/Supermarket	Sells the meat piece and recipes to consumers.	Product information • Price
Consumer	Buys the meat. Verifies origin, quality… of the meat they intend to buy. Fills in a form to be notified of product recalls (if needed)	Consumer information

Table 5: BC24-Trace blockchain participants.

Non-Hardware Event (NHE) indicates that data is manually inputted into the system rather than being retrieved from hardware sensors.

Material components

The Raspberry Pi is equipped with various components to facilitate its functionality within the supply chain system. It's important to consider these components because their security matters and they can influence the overall security of the system. They are:

Touchscreen Display: Users interact with the touchscreen to perform actions or visualize data. This interface provides an intuitive way to engage with the system and access information.

Breadboard: Extends the GPIO ports of the Raspberry Pi, enabling the addition of extra components.

RFID Reader: Used to read NFC tags.

Temperature Sensor: Monitors the storage conditions of meat throughout the supply chain journey, including during transportation, processing, and storage. It triggers alerts if the temperature exceeds predefined thresholds and records temperature data within the associated NFT for traceability purposes.

GPS Sensor: Verifies the location of animals at various checkpoints by comparing the GPS coordinates obtained during scanning with those stored in the NFC/NFT.

The integration of these components into the system is shown in the following architectural diagram (Figure 16).

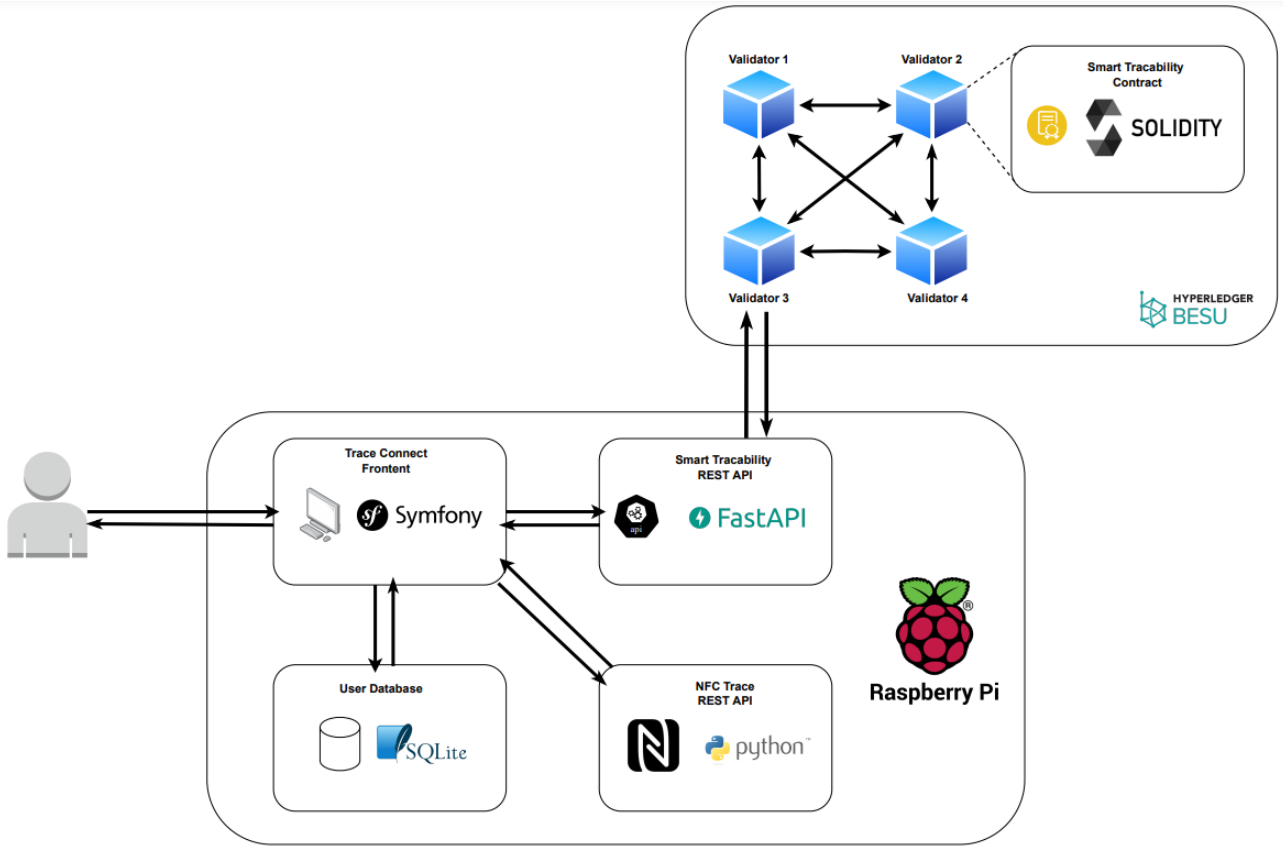


Figure 16: Architectural diagram of BC24-Trace system, from students of BC24-Trace case study.

2. Developing the case study

The process described above provides the basis for the development of the system. The presented development process is mainly focused on the blockchain-oriented extensions described in the previous sections.

i. Overall validation

a) Early requirements analysis

As mentioned earlier, the first phase of the Tropos methodology is the early requirements analysis. At this step, the designer will construct the security reference diagram, the actor diagram, and several goals diagrams. Figures 17, 18, and [19,20,21] present the diagrams. Regarding the proposed extensions, when creating these diagrams, the blockchain constructs retrieved from the alignment with Secure Tropos will be used. Additionally, the actor diagram will contain extensions for privacy and Norms concepts. In our BC24-Trace case study, we consider six key actors: the Farmer, the Transporter, the Butcher, the Factory, the Supermarket, and the Customers. (Refers to section “Overview of "BC24-Trace" Blockchain supply chain participants and process” for their description.)

The first step in the early requirements analysis as recommended by [13], is the construction of the security reference diagram. In our case, it is the blockchain security reference diagram, as mentioned in the proposal, in section “BOSE alignment with Secure Tropos”. The main security features of the blockchain security reference diagram for the BC24-Trace system are privacy, integrity, and availability.

Additionally, we add blockchain security features such as transparency and non-repudiation. We also consider consistency and partition tolerance, the C and P of the CAP theorem. CAP (Consistency, Availability, Partition tolerance) theorem is a fundamental theorem for defining transactional properties in distributed systems [33]. and in the context of a distributed ledger, the properties mean:

(1) Consistency: all nodes keep an identical ledger with most recent updates. (2) Availability: any transactions generated at any time in the network will be accepted in the ledger. (3) Partition tolerance: even if part of the nodes fails, the network can still operate normally [33].

The blockchain security reference diagram is shown in Figure 17.

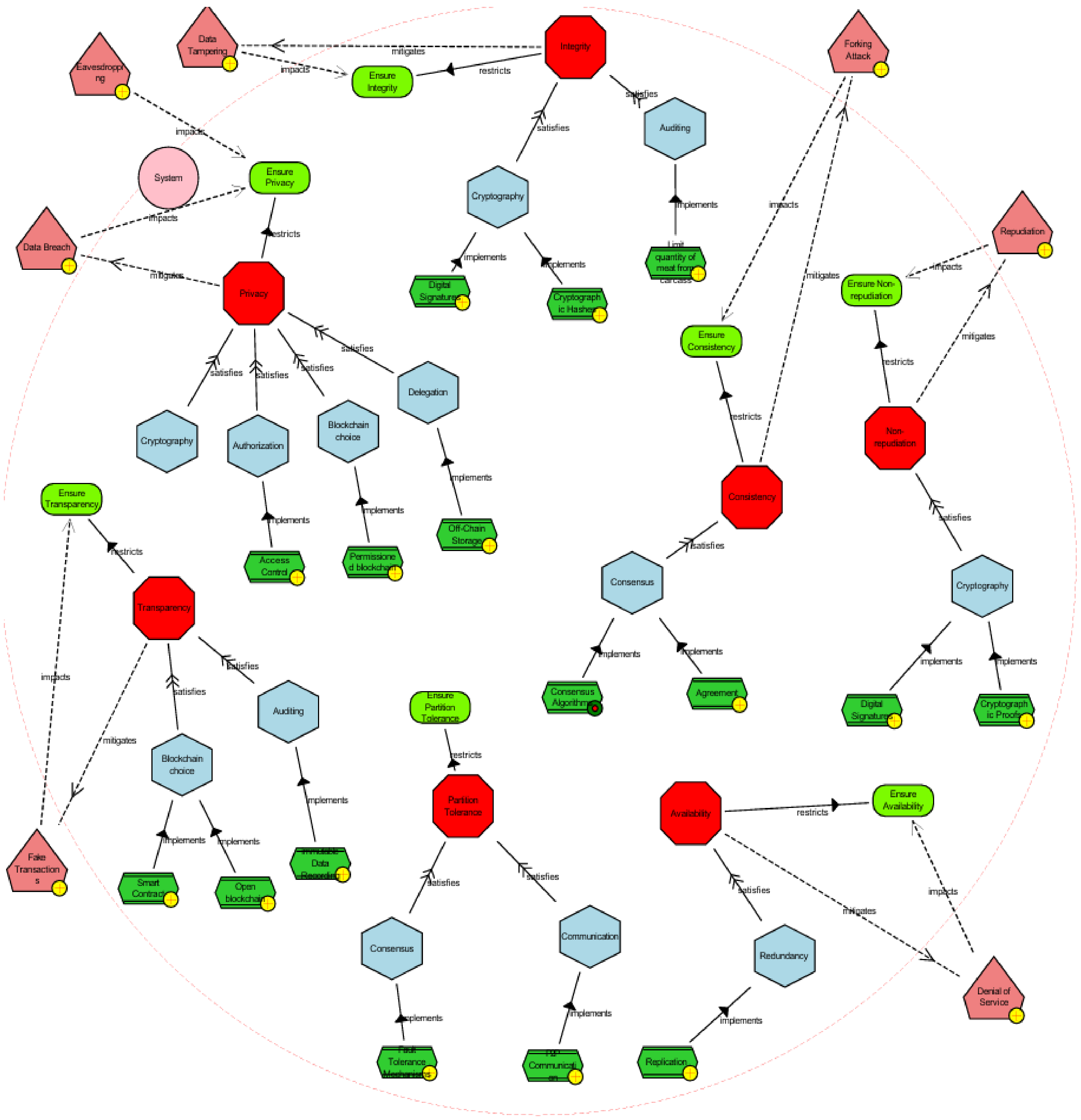


Figure 17:Blockchain Security Reference Diagram.

The next step of the process involves modelling the stakeholders of the system, along with their goals, dependencies, security constraints, and privacy considerations. For this purpose, the actor diagram is used. Once these elements are identified, we must conduct an in-depth analysis of each actor's goals and the security constraints imposed on them. In the following sections, I will cover these two steps together. My aim is not to demonstrate Secure Tropos methodology, but to present our extensions built upon it.

In the BC24-Trace case study, the farmer's goal is to raise the animals, tag NFC tags to them, and enter their breeding and health data into the blockchain. The farmer depends on the Transporter to move the animals from the farm to the butcher. This dependency needs a privacy consideration to restrict access to certain information to authorized persons only. For example, the Transporter does not need to see the food consumption and health information of the animals.

The transporter's goal is to move animals and meat products between different points in the supply chain and to log transportation details such as duration, temperature, and humidity conditions into the blockchain. The transporter relies on the farmer and butcher to collect the items he needs to transport, i.e. animals, carcasses, and meat pieces.

The butcher's goals are to process animals into meat carcasses, cut these carcasses into specific meat pieces, and record all processing data into the blockchain. He needs the transporter to get animals and to move processed meat carcasses to the factory. The butcher is imposed to comply with regulations like Corporate Sustainability Due Diligence (CS3D).

The factory's goals include preparing meat recipes, storing all manufacturing details in the blockchain, and packaging meat products according to supermarket requirements. Thus, it is obvious that the factory depends on the supermarket. Additionally, the factory depends on the transporter to receive meat carcasses, and to move packaged meat products to the supermarket. Security constraints for the factory include ensuring data confidentiality to protect proprietary recipes. They also need to conform with regulatory standards for food safety such as the European regulation about recalling (EC178/2002).

The supermarket's goal is to sell packaged meat products to customers, log product prices, and provide consumers with access to product origin and quality information. The Supermarket depends on the (Frozen) Transporter to receive packaged products.

Finally, the customer's goal is to purchase meat products and verify their origin, quality, and history. Therefore, they depend on the Supermarket to access this information. Ideally, this could be directly obtained if the Blockchain system was present however, this can be done in the next phase by delegating this task to the system actor. Consumers need the information to be always available and unbiased, imposing the following security constraints on the Supermarket: availability and integrity of product information. Privacy considerations involve providing consumers with access to only the necessary data to verify product quality and origin.

In addition to everything mentioned above, we can add access control for each actor to ensure that only authorized personnel can log the corresponding information. For example, only the Farmer can log animal health information, and only the Transporter can log transportation information, etc.

We noticed that each stakeholder needs to log information into the blockchain. However, during the early requirements analysis, the system actor is not yet modelled; this is done in the next phase. Therefore, we will not consider dependencies with the system at this moment. At that time, we can also add the constraint to ensure data integrity, which implies accurate logging of the correct data into the blockchain.

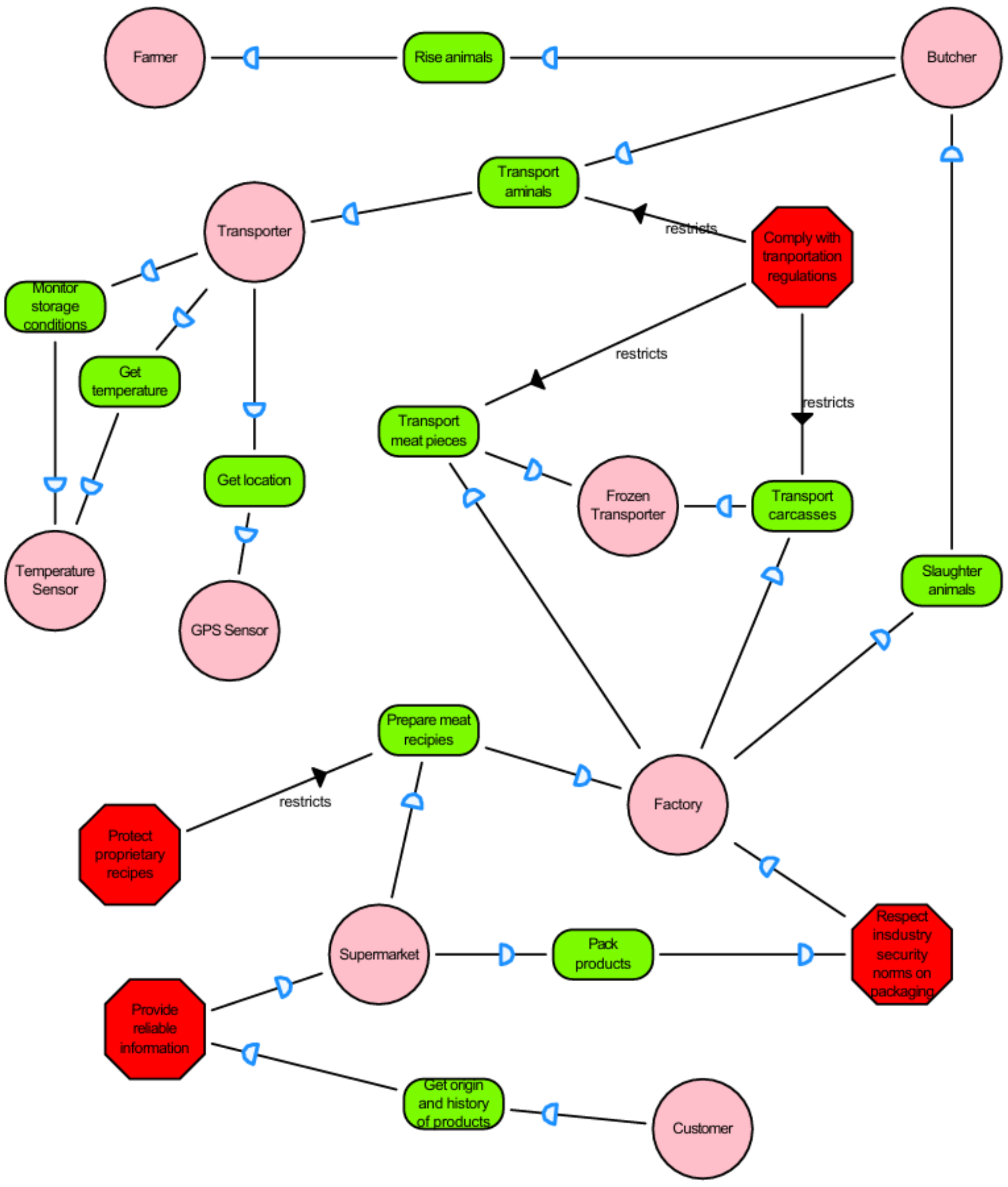


Figure 18: Actor diagram of BC24-Trace system (without system actor).

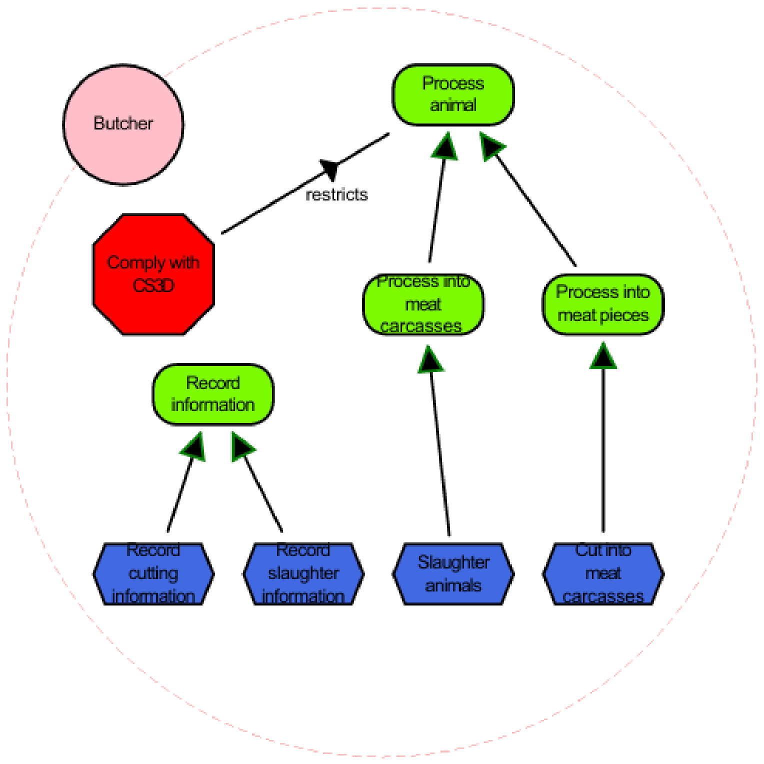


Figure 19: Goal diagram of butcher actor.

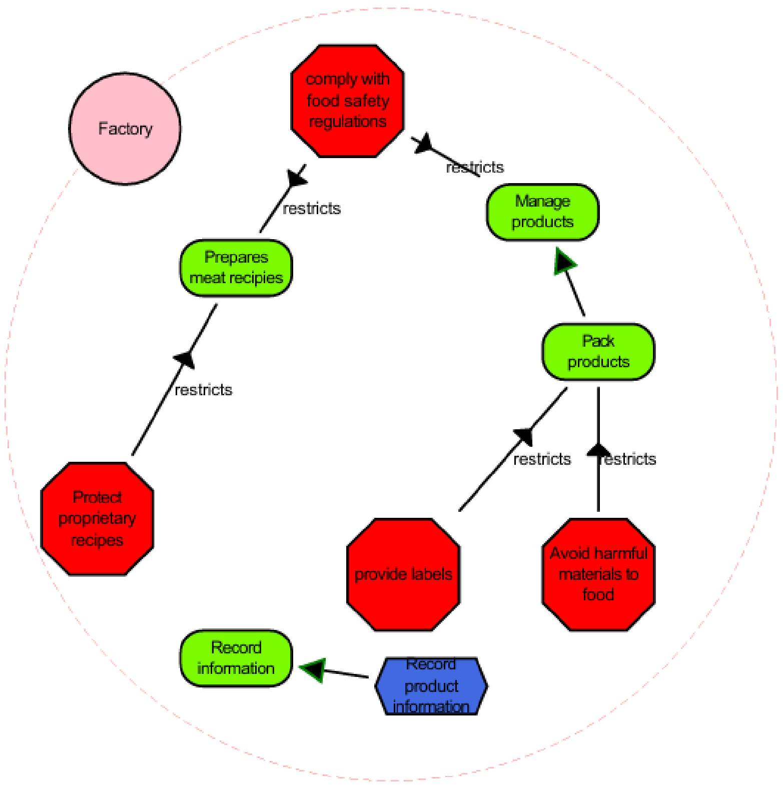


Figure 20: Goal diagram of Factory actor.

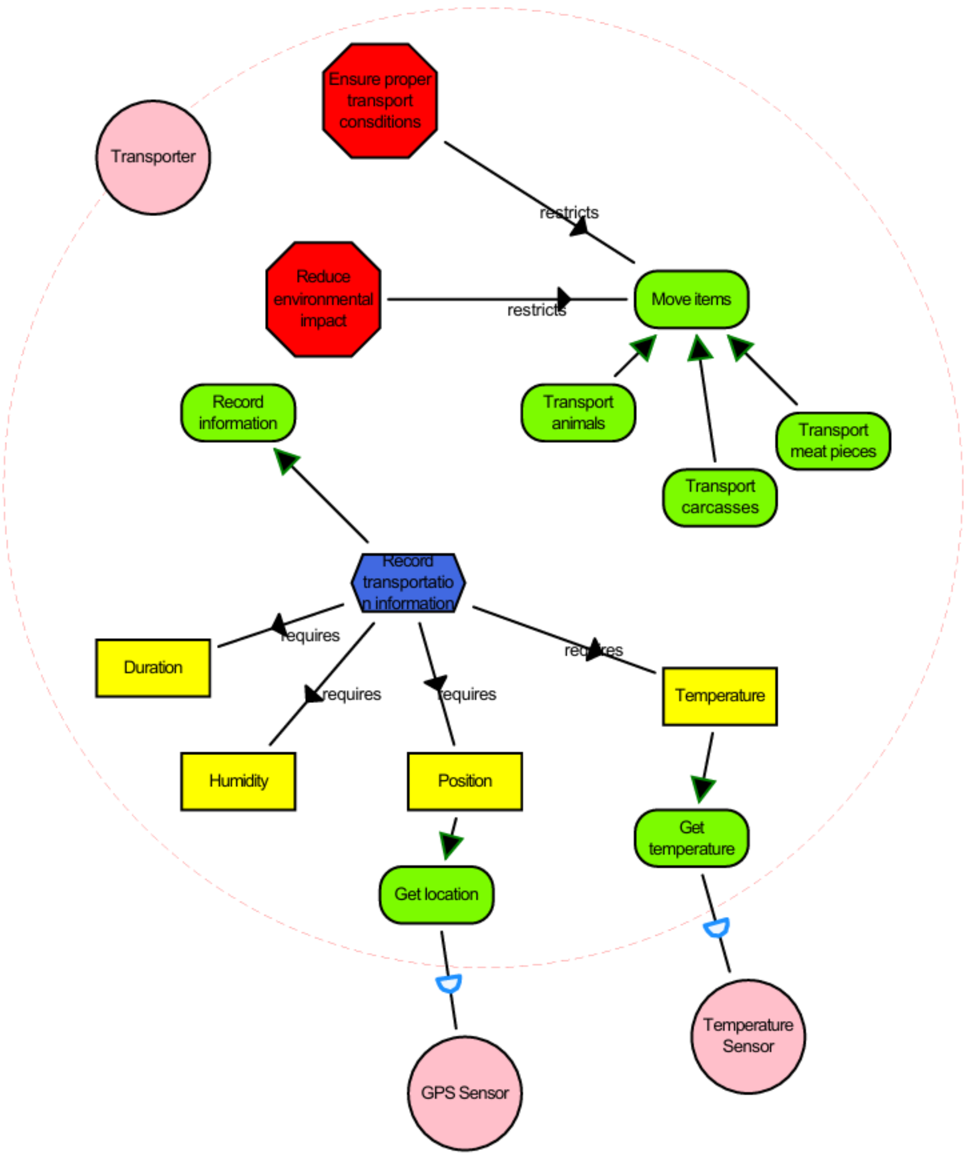


Figure 21: Goal diagram of transporter actor.

b) Late requirements analysis

During the late requirements stage, the system, BC24-Trace, is introduced as a new actor that has several dependencies with the other actors. For our extension, we will also add the smart contract actor because it is related to the system actor. New actors receive the responsibility for the fulfilment of some of the goals identified during the early requirements analysis that existing actors cannot satisfy. The designer constructs the actor diagram with the system and the goal diagram of the system. Figures 22 and 23 present an actor diagram with the system and a goal diagram of the smart contract actor. In the actor diagram with the system the dependencies of the system with the rest of the actors are the functional requirements of the system.

The main goal of the BC24-Trace system (see Figure 22) is to ensure traceability by storing data from stakeholders and the smart contract actor's goals are to Execute Contracts, Automate Transactions, and Ensure Compliance. Taking into consideration the blockchain security reference diagram (see

Figure 17), there are several main constraints imposed by the desired security features of the system: privacy, integrity, availability, transparency, non-repudiation, consistency, and partition tolerance. For the BC24-Trace system to satisfy these constraints, secure goals such as Ensure Data Privacy, Ensure Data Integrity, Ensure Data Availability, Ensure Data Transparency, Ensure Non-Repudiation, Ensure Data Consistency, and Ensure Partition Tolerance have been identified.

- For the privacy constraint, it can be further analysed into security sub-constraints: Allow Only Authorized Access, Allow Off-Chain Storage, and Decide on Blockchain Type. To address these, secure goals like Check Authorization, Manage Off-Chain Storage, and Choose Blockchain Type (e.g., permissioned blockchain) are introduced. For example, the Manage Off-Chain Storage goal can be divided into secure tasks such as Store Data Off-Chain and Retrieve Data Off-Chain. We could go deeper by specifying the type of off-chain storage, such as using a side chain, but we avoid doing so at this stage to give system designers flexibility in implementation.
- For the integrity security constraint, secure goals like Use Cryptographic Hashes, Verify Digital Signatures, and Conduct Audits are identified.
- For availability, secure goal like Replicate Data is defined.
- For transparency, secure goals such as Implement Smart Contracts and Choose Blockchain Type are identified.
- For non-repudiation, the secure goals are Use Digital Signatures and Provide Cryptographic Proofs.
- For consistency, consensus algorithms are used to ensure all nodes agree on the state of the blockchain. The secure goal here is Apply Consensus Algorithms.
- For partition tolerance we can implement fault tolerance mechanisms and ensuring P2P communication. The corresponding secure goals are Implement Fault Tolerance Mechanisms and Ensure P2P Communication.

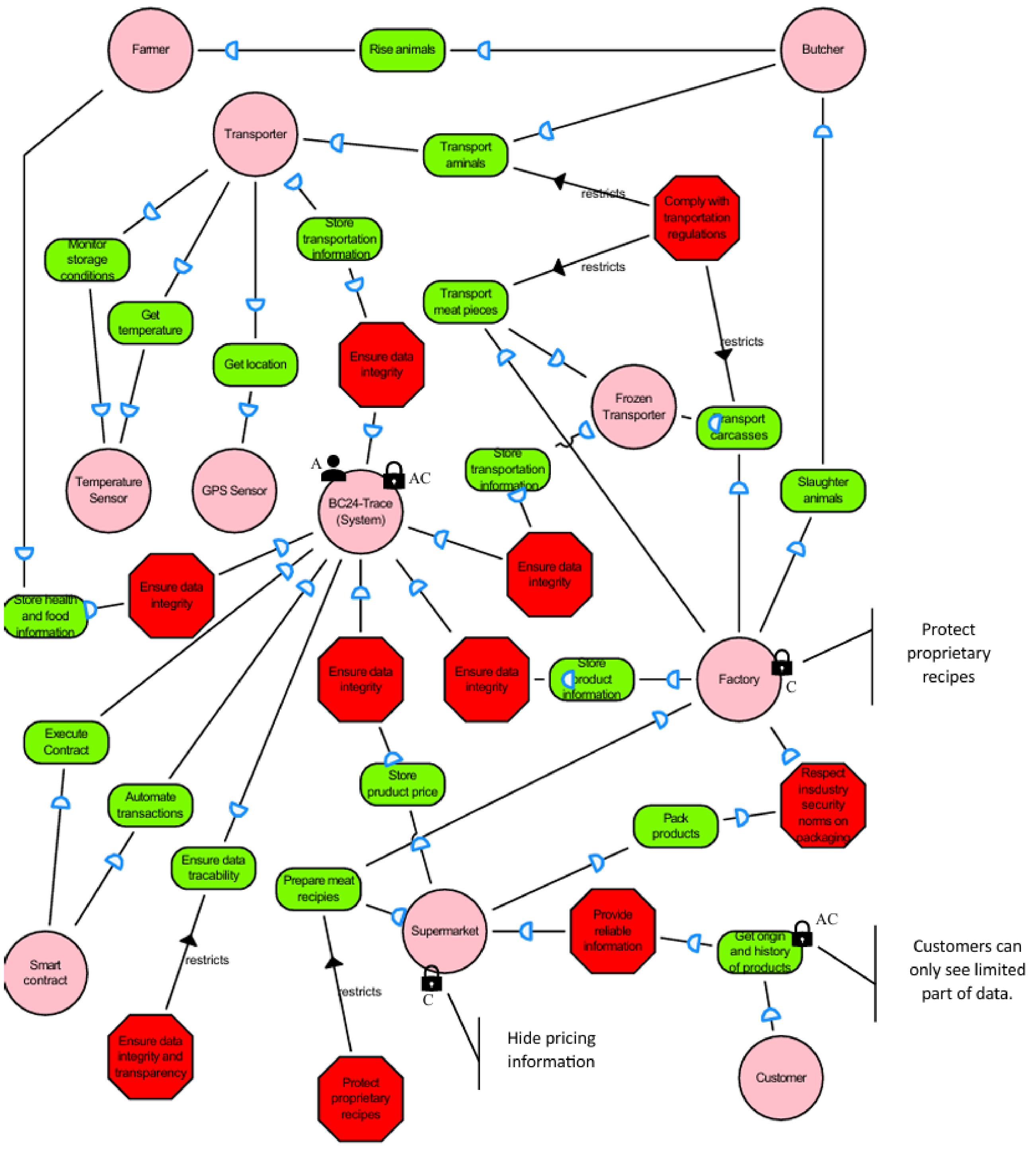


Figure 22: Actor diagram with the system.

Figure 22 also illustrates the integration of privacy considerations, specifically focusing on Access Control, Confidentiality and Anonymity. Consumers are restricted to see only a limited part of the blockchain data through the implementation of access control. The system enforces anonymity for certain data elements to protect privacy. System also implements access control to ensure that only supply chain participants can access the blockchain. Finally, the factory actor implements confidentiality to his protect proprietary recipes, and the supermarket actor hides pricing information to maintain confidentiality.

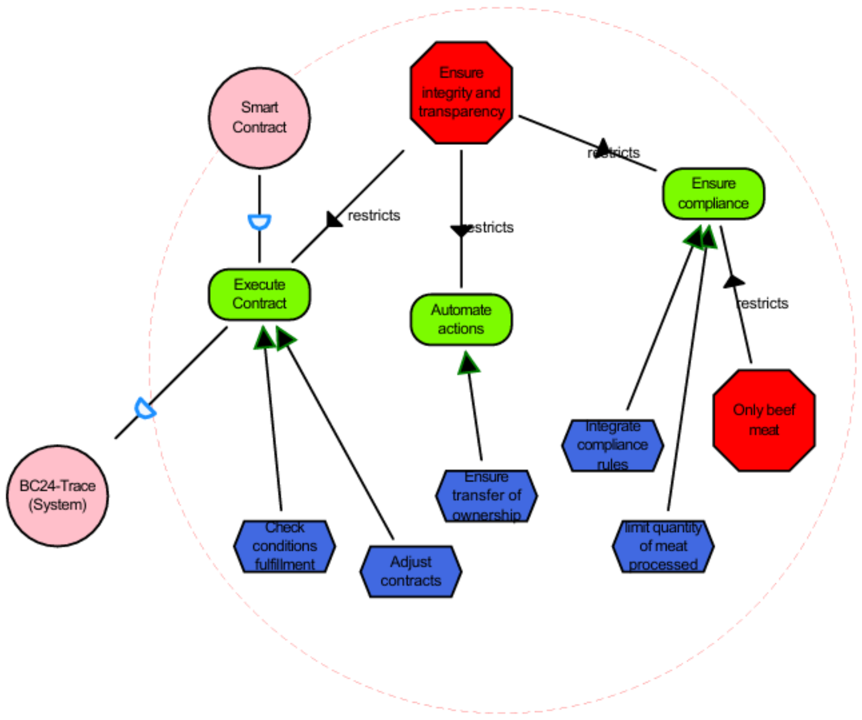


Figure 23: Goal diagram of smart contract actor.

On the other hand, the smart contract actor's goals are to Execute Contracts, Automate Transactions, and Ensure Compliance. The constraints imposed on it, in addition to those previously mentioned, are the requirements of smart contract testing (an extension).

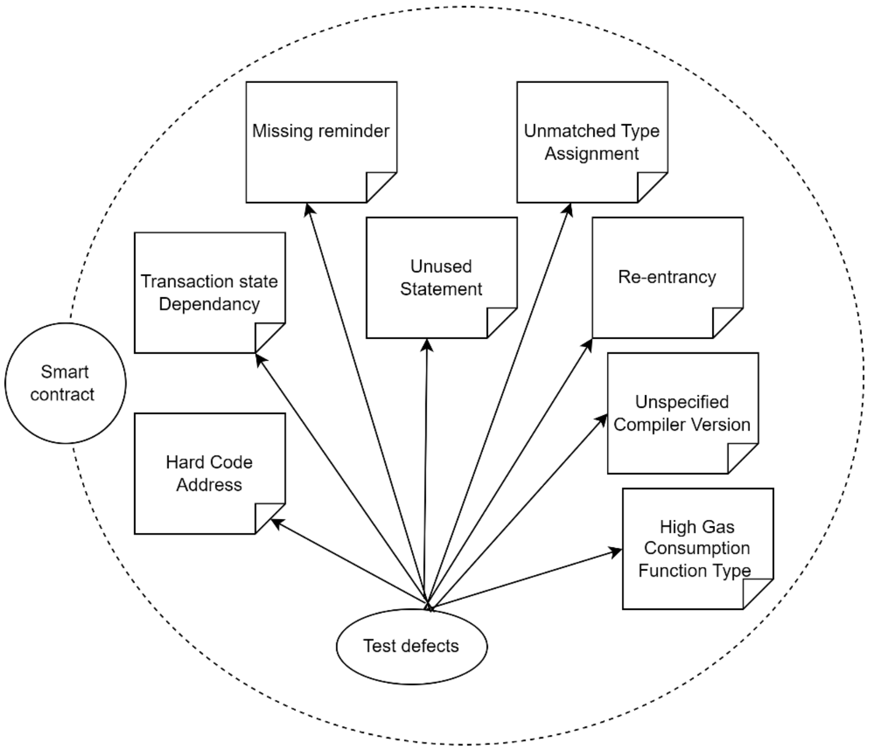

Missing reminder
Unmatched Type Assignment
Transaction state Dependancy
Unused Statement
Re-entrancy
Smart contract
Unspecified Compiler Version
Hard Code Address
High Gas Consumption Function Type
Test defects

Figure 24: Smart contract actor with his testing criteria.

To avoid overwhelming the reader and since all the concepts related to our extension and applicable here have been presented, the Architectural Design and Detailed Design sections will not be included. However, they follow the same procedures as in secure Tropos. As a reminder:

In the architectural design step, the system's architecture is defined. This involves, the actor decomposition, system actor diagram refinement, Capabilities identification and Agent assignment. (Refers to section "Tropos methodology" for more information).

For the detailed design, the developer specifies the agent capabilities and interactions, considering the security aspects derived from the previous steps of the analysis. (Refers to section "Tropos methodology" for more information).

ii. Validation of 'Norm & laws' concept

Presentation of the Corporate Sustainability Due Diligence Directive

The CS3D, adopted on April 24, 2024, aims to make companies act responsibly by addressing human rights and environmental impacts in their business activities. This directive requires large EU companies, and some non-EU companies with significant business in the EU, to identify, prevent, mitigate, and account for negative impacts on people and the planet within their operations, subsidiaries, and supply chains. It supports the European Green Deal and the UN Sustainable Development Goals, promoting a transition to a climate-neutral and green economy. Companies must develop due diligence policies, monitor their supply chains, and establish grievance mechanisms. If they don't comply, they can face fines and liability for damages. While the directive mainly targets large corporations, it also indirectly supports SMEs through business relationships. By setting high standards for corporate behaviour, the directive encourages transparency, accountability, and sustainable practices, contributing to a fairer and more environmentally friendly economy [34].

'Norm & laws' validation

The BC24/Trace project is directly relevant to the CS3D because it operates within the food industry and involves a comprehensive supply chain. As mentioned above, the directive requires companies, especially in the food sector, to identify, prevent, mitigate, and account for adverse human rights and environmental impacts throughout their operations and supply chains. Our project, which guarantees a full transparency in the beef supply chain using blockchain technology, must therefore take account of the directive's requirements. It tracks animal health, treatment, transport conditions, and ownership transfers, providing consumers with clear information on product origin and quality.

Additionally, part II of the directive's annex addressing the violation of internationally recognized objectives and environmental violations, mentions the prohibition of importing or exporting endangered species without a permit, as stated in the Convention on International Trade in Endangered Species of Wild Fauna and Flora (CITES) [35]. Therefore, our project must prove that we neither import nor use endangered animals, such as giraffes, in our supply chain. And the proposed methodology achieves this by using the concept of laws and norms. This capability can be used to show that our system prevents the import of endangered species, thus justifying that our solution can be used as a legal basis for proving compliance with due diligence obligations.

In fact, BC24/Trace project’s supply chain can prevent illegal activities such as mislabelling or smuggling endangered species like giraffes due to multiple layers of checks throughout the process. At the beginning of the chain, the likelihood of substitution fraud, such as replacing a beef with a giraffe, is extremely low due to the initial checks by the transporter. If a farmer inaccurately tags a giraffe as a beef, the transporter, will immediately notice the discrepancy due to the obvious differences in appearance and characteristics between these species. This initial human verification ensures that only beef enters the supply chain. Even if the transporter colludes in the fraud, it is still possible to remark it at later stages, such as at the butcher’s or when the carcass arrives at the factory. Furthermore, even if the factory wishes to participate in this fraudulent activity, it will be strongly discouraged from doing so, as such behaviour would expose them to severe legal responsibilities and risks, including fines, reputational damage, and potential criminal charges. The same applies to farmers, transporters, and butchers, who must maintain the integrity of their part of the supply chain, as their involvement in fraud would breach regulatory compliance and undermine their business operations due to legal and economic repercussions.

Another potential fraud could occur at the factory, where there might be an attempt to mix beef from an uncorrupted supply chain with illegally obtained giraffe meat. This scenario mirrors the case of the French brand Spanghero, which found horsemeat in its "pure beef" products. A meat trader imported horsemeat from Romania and Canada, removed labels in Dutch warehouses, and sold it to Spanghero in France, where it was mislabelled as beef. Our system addresses this problem by ensuring that the meat processed and sold is exactly as recorded and traced through the blockchain. The fundamental idea is to provide undeniable proof that the meat handled by our butchers comes from legitimate sources and is not mislabelled giraffe meat.

In fact, the system limits the quantity of meat that can be produced from a carcass and verifies if the quantity of meat labelled by the factory matches the number of carcasses used. For example, if we have 10 beef carcasses and each is expected to yield 10 kilograms of meat, the system ensures that no more than 100 kilograms of minced meat can be processed. This is enforced through smart contracts in the blockchain that prevent recording any amount exceeding the original yield. This system inherently discourages mixing different types of meats. If any additional meat, such as giraffe meat, were added illegally, the excess meat could not be sold through legal channels as part of the traced supply without exposing the discrepancy between expected and actual production. Any untraceable meat exiting the factory would fetch a lower price due to the inability to prove its origin, posing legal risks and diminishing its economic value.

To align more closely with legal requirements, the project could integrate waste management by ensuring that meat waste (such as beef bones…) is transferred to certified companies for proper treatment. Smart contracts will automatically enforce environmental regulations, ensuring waste is handled only by approved entities under specific conditions, with blockchain records verifying compliance. Additionally, regular inspections and audits of the waste management process could be conducted, storing inspection reports and audit results on the blockchain for review by regulators and stakeholders, confirming adherence to environmental best practices.

Since all specifications from the CS3D directive were translated with the concept of "law and norm" (see Figure 25), it was simpler to determine what needed to be implemented in the system to be compliant and to prove the legality of our operations.

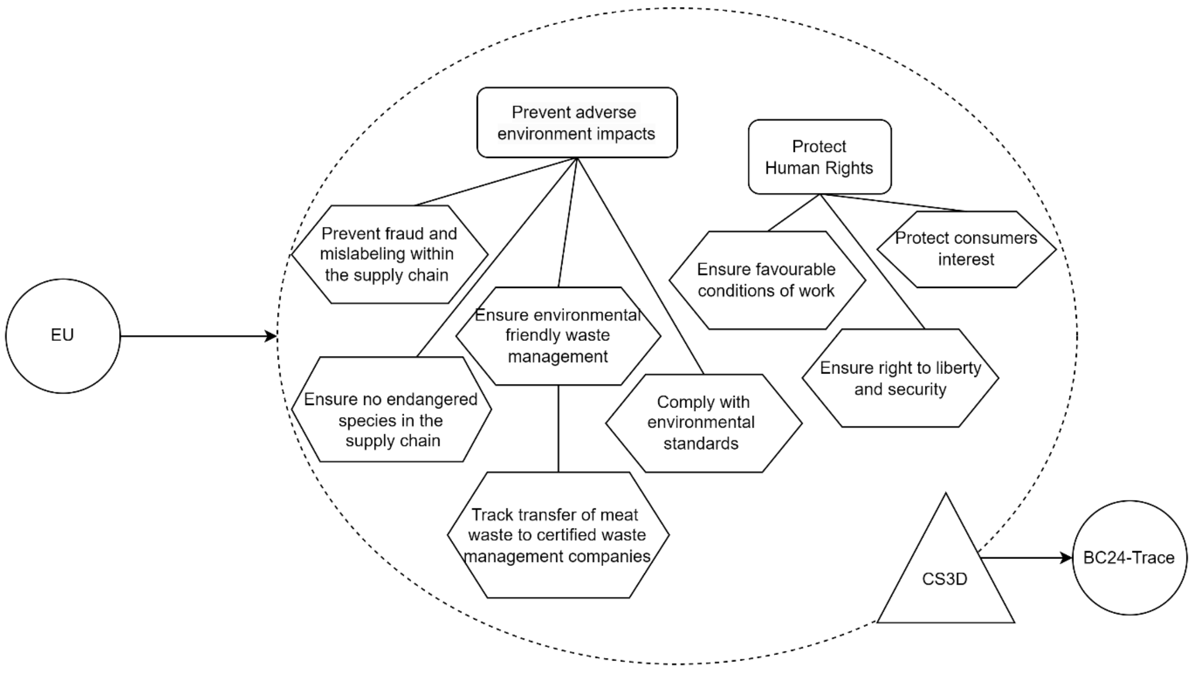

Prevent adverse environment impacts
Protect Human Rights
Prevent fraud and mislabeling within the supply chain
Protect consumers interest
Ensure favourable conditions of work
EU
Ensure environmental friendly waste management
Ensure right to liberty and security
Ensure no endangered species in the supply chain
Comply with environmental standards
Track transfer of meat waste to certified waste management companies
CS3D
BC24-Trace

Figure 25: CS3D Directive representation with "Norm and law" concept.

iii. Validation of 'Decision link' concept

Before diving into the concept of "Decision Link," let's quickly remind its context. This idea is about two laws with the same goal but approaches that might conflict, requiring a system manager to make a decision on how to prioritize and mitigate the problem.

To illustrate this, let's look at two specific laws: the EU Animal Welfare Regulation and the Canadian Corporate Respect for Human Rights and the Environment Act.
The EU Animal Welfare Regulation, proposed in December 2023, aims to ensure the humane treatment and protection of animals during transport. Meanwhile, the Canadian Corporate Respect for Human Rights and the Environment Abroad Act focuses on making Canadian companies operating overseas prevent and address any harm they might cause to people or the environment. This act mandates companies to implement due diligence procedures to avoid harm and holds them accountable if they fail to do so [36].

Both laws are relevant to the Trace project as they aim to improve ethical standards in animal transportation but in potentially conflicting ways. The EU regulation emphasizes humane transport conditions, including specific requirements for rest periods, temperature control, and space allowances. In contrast, the Canadian law prioritizes reducing the environmental impact of animal transportation, with a focus on lowering emissions and energy consumption.

The conflict arises from several specific elements:

- Firstly, the EU Animal Welfare Regulation mandates regular rest periods during transportation to ensure animal welfare. However, these longer rest periods mean more frequent stops, which can increase the total travel time and fuel consumption. This directly conflicts with the Canadian law's emphasis on minimizing stops to reduce emissions.

- Secondly, the EU regulation requires maintaining specific temperature conditions during transport, often necessitating climate-controlled vehicles. These vehicles consume more energy, increasing the carbon footprint, which contradicts the Canadian goal of reducing energy consumption and emissions.
- Lastly, the EU specifies minimum space requirements per animal to ensure comfort, potentially leading to fewer animals being transported per vehicle. This results in more trips and higher overall fuel consumption, opposing the Canadian law's objective to maximize load efficiency and reduce trips and emissions.

These conflicting requirements present challenges, such as increased costs and the legal risks. The "Decision Link" helps address these conflicts by allowing designers to visually identify potential problems early in the design phase and plan their resolution.

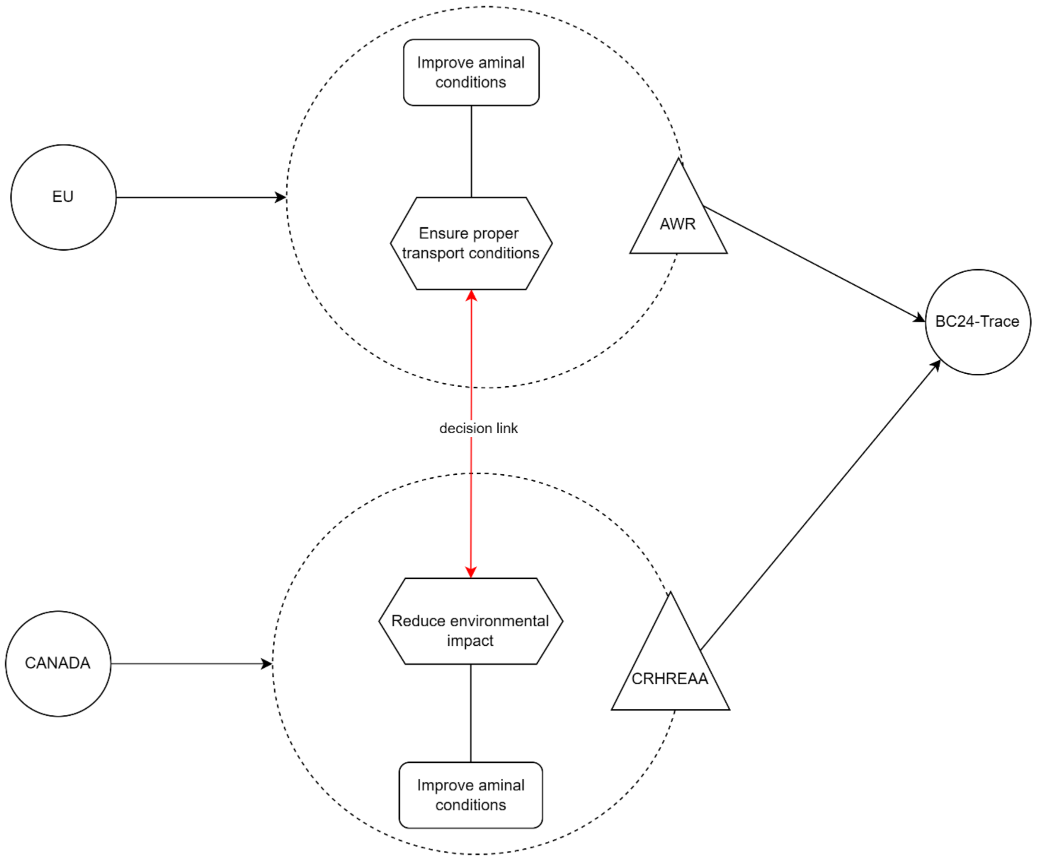

Improve aminal conditions
EU
Ensure proper transport conditions
AWR
BC24-Trace
decision link
CANADA
Reduce environmental impact
CRHREAA
Improve aminal conditions

Figure 26: "Decision link" concept validation part 1.

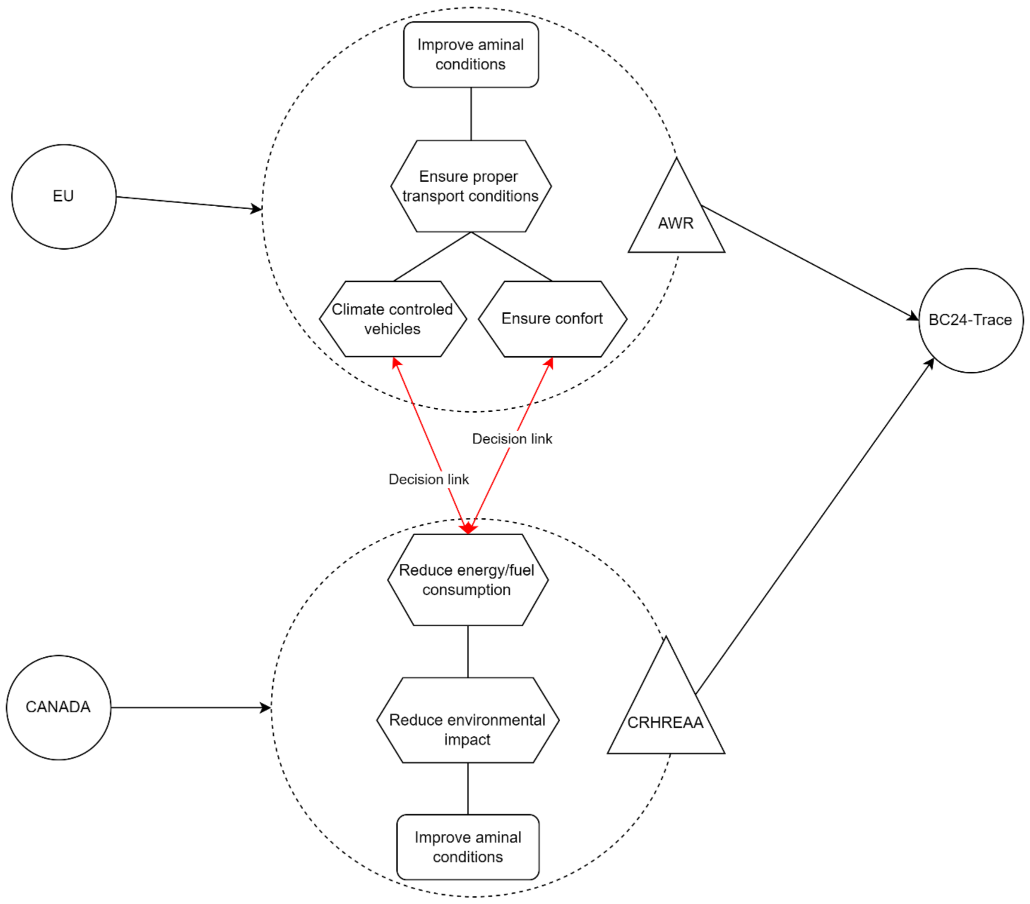

Improve aminal conditions
Ensure proper transport conditions
EU
AWR
BC24-Trace
Climate controled vehicles
Ensure confort
Decision link
Decision link
Reduce energy/fuel consumption
CANADA
Reduce environmental impact
CRHREAA
Improve aminal conditions

Figure 27: "Decision link" concept validation part 2 (more detailed).

iv. Validation of 'Testing criteria' concept

Validation of this concept will be done at the final step of the project, just before deployment and will take the form of a conformance checking of the smart contract tests. Specifically, I will ask the developers of the BC24-Trace project to show the tests they have conducted. I will then verify whether all the testing criteria proposed in the model have been met. It is important to remember that the model already includes potential problems and defects that could compromise the security of the smart contracts, which is reflected by the testing criteria.

After analyzing the smart contract and its corresponding tests in relation to the defects chosen for the BC24-Trace system (Figure 24), we can conclude the following:

- The ***Transaction State Dependency defect*** has been respected. The contract does not use "tx.origin", and in the tests, it is not used to verify permissions as required by the defect description (see Figure 14). Instead, the developers have implemented a role-based access control to ensure that permissions are correctly assigned and validated. This is demonstrated in tests such as "should assign breeder role correctly" or "should not allow non-admin to assign roles."
- The ***Hard Code Address defect*** has been respected. The contract uses dynamic role assignment rather than hard-coded addresses.
- The ***Unspecified Compiler Version defect*** has not been respected. In the smart contract, we can see that the compiler version is specified to "pragma solidity ^0.8.20", meaning it is can

use solidity version 0.8.20 or higher. The defect specification requires setting a specific version to minimize security risks related to unexpected changes in the compiler. A good practice would be to set "pragma solidity 0.8.20".

- The ***Unused Statement defect*** has been respected. The contract logic appears clean without redundant code or unused values. This type of defect can be detected directly within the development environment used.
- The ***Missing Reminder defect*** seems to be respected. Events are emitted to notify about state changes, such as resource creation and metadata changes. This can be seen in tests like "should allow breeder to mint sheep resource" where it is verified that "ResourceCreatedEvent" is called when the "mintRessource" function is executed by a user with the breeder role.
- The ***High Gas Consumption Function Type defect*** has not been explicitly tested. The test suite does not include tests specifically measuring gas consumption or verifying if the appropriate function type has been used as described in the defect (see Figure 14). It is important to test this to avoid high gas costs.
- The ***Unmatched Type Assignment defect*** has not been tested but seems to be respected. There are no apparent issues with unmatched type assignments in the smart contract, but to be more confident, we should test it. For example, we could include test cases that deliberately misuse types to confirm that the contract handles the situations correctly and reverts as expected.
- The ***Re-entrancy defect*** has not been tested, which is not very alarming as the smart contract does not make real external calls. However, functions involving minting (mintRessource) and burning (_burnResources) could potentially benefit from re-entrancy protection. Even if the functions called are part of the ERC1155 standard and are safe from re-entrancy when used correctly, implementing re-entrancy protection would be beneficial, especially as OpenZeppelin offers this functionality with "ReentrancyGuard" and "nonReentrant".

A summary of the analysis of smart contract and tests on smart contract is presented in the table 6.

Contract defect	Respected?	Verified? / Tested?	How it is tested
Transaction State Dependency	Yes	Yes	Role-based access control tests
Hard Code Address	Yes	Yes	Role assignment tests
Unspecified Compiler Version	No	Yes (verified)	Directly in the smart contract
Unused Statement	Yes	Yes (verified)	The development environment and test coverage
Missing Reminder	Yes	Yes	Event emission tests
High Gas Consumption Function Type	Maybe	No	-
Unmatched Type Assignment	Maybe	No	-
Re-entrancy	Maybe	No	-

Table 6: Summary of smart contract tests analysis

VII. DISCUSSION

The results of the case study demonstrated that we could efficiently model a blockchain-based system and extract its requirements (security, legal, privacy…) using the proposed Secure Tropos extension. Indeed, the approach helps towards the development of secure blockchain systems in many points:

The primary advantage is the introduction of specific constructs and notations tailored for blockchain components, such as blockchain actors (e.g., nodes, wallets, smart contracts). This is a significant step forward in blockchain modelling, as it provides appropriate elements with roles and interactions that reflect their functions within a blockchain network. As a result, developers are not disoriented and will not have to translate concepts that might confuse them. This uniformity in development leads to a better definition of blockchain systems, ensuring comprehensive coverage of all relevant aspects.

Another advantage is the ability to integrate blockchain security considerations from the early stages of the system development process. By integrating security features directly related to blockchain, such as partition tolerance and non-repudiation, the methodology allows for the identification of desired security requirements very early. And, these requirements are propagated throughout the development process, ensuring that security is always considered. For example, in the case study, during the early requirements analysis, we identified BC security features like Privacy, Integrity, Availability, Consistency, Partition Tolerance, Non-repudiation, and Transparency when creating the blockchain security reference diagram. It was from these attributes we derived the system's secure goals during the late requirements analysis.

The methodology also provides robust support for privacy concerns through constructs like access control and accountability…. This enables developers to clearly define the privacy requirements of the system. This can be seen in the case study when the Farmer restricted access to the food consumption and health information of the animals to the Transporter.

Legal and regulatory compliance is another aspect addressed by the proposed approach. The methodology can translate legal requirements, into clear legal requirements. This was well-demonstrated in the validation phase, where we translated and represented CS3D requirements into specific compliance goals in the system design. Legislation is vital for blockchain systems, as they will quickly face many legal requirements. Incorporating these into the methodology ensures that the systems developed are legally compliant and meet regulatory standards.

Then there's the fact that the method can address (timidly for the moment), a problem that affects many blockchain systems: the security of smart contracts. By requiring tests of smart contracts against potential attacks and known defects (testing criteria), developers can validate the security of these contracts and, by extension, the entire blockchain system and reduce the risk of costly post-deployment patches. For example, during the validation phase of the case study, the model helped detect several defects in the smart contract that were not explicitly tested or verified by the developers. For instance, the compiler version was not specified to a specific version, which could increase security risks. We also noted the lack of tests for unmatched type assignments, which can lead to issues like integer overflows. Additionally, there was no implementation of re-entrancy protection for functions involving minting and burning resources. Ultimately, the validation of the testing criteria clearly showed that some defects were neglected (Unmatched Type Assignment, Re-entrancy) or even forgotten. Since tests and verification are not done, we cannot say with certainty that we are safe or not, highlighting the importance of thorough testing criteria.

Finally, the integration of simple, well-known concepts, and the successful incorporation of the approach within the development stages of the Secure Tropos methodology, contribute to a clear and well-guided security-oriented process. This allows even non-security-oriented developers to consider security in their design.

VIII. CONCLUSION

In this thesis, we have presented an extension of the Secure Tropos methodology to build blockchain-based software systems. We proposed several contributions. First, we identified the key aspects of BBS that require special consideration during the development process and the actual gaps in Secure Tropos. Second, we developed the extension that integrates blockchain-specific constructs into the Secure Tropos framework, as well as privacy concepts, legal compliance elements, smart contract testing criteria, and detailed security mechanisms. Finally, we demonstrated the effectiveness of our extension through the BC24-Trace case study.

1. Limitations

One limitation of this enhanced Secure Tropos methodology concerns the list of 20 defects on Ethereum proposed for the testing criteria. Focusing on Ethereum at the moment is a limitation, but since we wanted to apply our results to the BC24-Trace project, which uses Ethereum, it was better to start with this. However, since this is a new approach, we leave it to the community to test and identify other possible limitations.

2. Answers to research questions

To answer my main research question, which was: "**How to extend the Secure Tropos methodology to build BBS?**", we will answer the three sub-questions that were asked.

1. What aspects of BBS should be considered during its development?

The blockchain modelling perspectives that should be considered during its development are the same as those for a typical system: Structure, Process, and Data. For this master's thesis, extensions have been proposed at all levels.

2. How can Secure Tropos methodology implement Blockchain principles?

While analysing Secure Tropos concepts, we identified gaps in the methodology regarding the development of BBS, such as failure to capture blockchain security attributes, inadequate modelling of blockchain components, and limited support for security evaluation and testing. However, thanks to Secure Tropos' flexibility, we have made several additions throughout this master’s thesis. As a result, the Secure Tropos methodology can implement blockchain principles through the followings: blockchain entities (CB actors, CB resources, CB goals, CB tasks, CB capabilities, CB security features, CB protection goals and CB security mechanisms), norms and laws, decision link concept, privacy considerations and smart contract testing requirements.

3. How to apply enhanced Secure Tropos methodology to BBS development?

To apply the proposed Secure Tropos extension to BBS development, we follow the same process as in Secure Tropos: early requirements analysis, late requirements analysis, architectural design, detailed design and implementation. However, additional activities are added during these stages. Refers to section “Application Method” for more details.

3. Future work

The proposed extensions are more focused on structure and process aspects of a system development. This is quite normal as it is the first attempt, and it is important to establish the foundations before tackling more complex parts. Therefore, in the future, we plan to propose more concepts that address the data aspect.

Another possibility for future work concerns the smart contract testing already proposed in this thesis. As mentioned in the limitations section, the list of defects provided is not exhaustive compared to what is observed in the field. We could therefore expand it with a more complete list of defects, not only concerning Ethereum but smart contracts in general.

Continuing with smart contracts, we could consider automatic test generation and integrating patterns/anti-patterns in the future to simplify developers' tasks.

Finally, we could develop a modelling tool that includes the presented extension, but it's too early to consider it at the moment.

IX. BIBLIOGRAPHY

[1] Faruk, Md Jobair Hossain, et al. "Software engineering process and methodology in blockchain-oriented software development: A systematic study." 2022 IEEE/ACIS 20th International Conference on Software Engineering Research, Management and Applications (SERA). IEEE, 2022.

[2] Belotti, Marianna, et al. "A vademecum on blockchain technologies: When, which, and how." IEEE Communications Surveys & Tutorials 21.4 (2019): 3796-3838.

[3] Bashir, Imran. Mastering blockchain. Packt Publishing Ltd, 2017.

[4] Bitcoin: A Peer-to-Peer Electronic Cash System

[5] Aslam, Sidra, Aleksandar Tošić, and Michael Mrissa. "Secure and privacy-aware blockchain design: requirements, challenges and solutions." Journal of Cybersecurity and Privacy 1.1 (2021): 164-194.

[6] Chaganti, Rajasekhar, Bharat Bhushan, and Vinayakumar Ravi. "A survey on Blockchain solutions in DDoS attacks mitigation: Techniques, open challenges and future directions." *Computer Communications* 197 (2023): 96-112.

[7] Levasseur, Olivier, Mubashar Iqbal, and Raimundas Matulevičius. "Survey of model-driven engineering techniques for blockchain-based applications." Proceedings http://ceur-ws. org ISSN 1613 (2021): 0073.

[8] Porru, Simone, et al. "Blockchain-oriented software engineering: challenges and new directions." 2017 IEEE/ACM 39th International Conference on Software Engineering Companion (ICSE-C). IEEE, 2017.

[9] Destefanis, Giuseppe, et al. "Smart contracts vulnerabilities: a call for blockchain software engineering?." 2018 International Workshop on Blockchain Oriented Software Engineering (IWBOSE). IEEE, 2018.

[10] Vacca, Anna, et al. "A systematic literature review of blockchain and smart contract development: Techniques, tools, and open challenges." Journal of Systems and Software 174 (2021): 110891.

[11] Mouratidis, Haralambos. A security-oriented approach in the development of multiagent systems: applied to the management of the health and social care needs of older people in England. Diss. University of Sheffield, 2004.

[12] Matulevičius, Raimundas. Fundamentals of secure system modelling. Springer, 2017.

[13] Mouratidis, Haralambos, and Paolo Giorgini. "Secure tropos: a security-oriented extension of the tropos methodology." International Journal of Software Engineering and Knowledge Engineering 17.02 (2007): 285-309.

[14] http://www.troposproject.eu/files/8-Tropos-Basics.pdf

[15] Bresciani, Paolo, et al. "Tropos: An agent-oriented software development methodology." Autonomous Agents and Multi-Agent Systems 8 (2004): 203-236.

[16] Curran, Kevin, Amanda Millar, and Conor Mc Garvey. "Near field communication." International Journal of Electrical and Computer Engineering 2.3 (2012): 371.

[17] Wang, Qin, et al. "Non-fungible token (NFT): Overview, evaluation, opportunities and challenges." arXiv preprint arXiv:2105.07447 (2021).

[18] https://en.wikipedia.org/wiki/Raspberry_Pi

[19] Vingerhouts, Anne Sofie, Samedi Heng, and Yves Wautelet. "Organizational modeling for blockchain oriented software engineering with extended-i* and UML." CEUR Workshop Proceedings. Vol. 2749. CEUR Workshop Proceedings, 2020.

[20] Shi, Yani, et al. "Service-oriented modeling for blockchain-enabled supply chain quality information systems." Security and Communication Networks 2022 (2022).

[21] Çelebi, Ilhan. Privacy enhanced secure tropos: A privacy modeling language for gdpr compliance. Diss. Master Thesis, 2018.

[22] Rek, Patrik, and Muhamed Turkanović. "Data modelling for blockchain oriented software engineering." Central European Conference on Information and Intelligent Systems. Faculty of Organization and Informatics Varazdin, 2021.

[23] Kitchenham, Barbara, et al. "Systematic literature reviews in software engineering–a systematic literature review." Information and software technology 51.1 (2009): 7-15.

[24] Elliott, Steve. "Proof of Concept Research." Philosophy of Science 88.2 (2021): 258-280.

[25] Pavlidis, Michalis, Shareeful Islam, and Haralambos Mouratidis. "A CASE tool to support automated modelling and analysis of security requirements, based on secure tropos." IS Olympics: Information Systems in a Diverse World: CAiSE Forum 2011, London, UK, June 20-24, 2011, Selected Extended Papers 23. Springer Berlin Heidelberg, 2012.

[26] Islam, Shareeful, Haralambos Mouratidis, and Jan Jürjens. "A framework to support alignment of secure software engineering with legal regulations." Software & Systems Modeling 10.3 (2011): 369-394.

[27] Aggarwal, Shubhani, and Neeraj Kumar. "Blockchain components and concepts." Advances in Computers. Vol. 121. Elsevier, 2021. 387-398.

[28] Siena, Alberto, et al. "Exploring the effectiveness of normative i* modelling: Results from a case study on food chain traceability." Advanced Information Systems Engineering: 20th International Conference, CAiSE 2008 Montpellier, France, June 16-20, 2008 Proceedings 20. Springer Berlin Heidelberg, 2008.

[29] Siena, Alberto, et al. "Designing law-compliant software requirements." Conceptual Modeling-ER 2009: 28th International Conference on Conceptual Modeling, Gramado, Brazil, November 9-12, 2009. Proceedings 28. Springer Berlin Heidelberg, 2009.

[30] Hamadi, Yasmine Ben, Samedi Heng, and Yves Wautelet. "Using i*-based organizational modeling to support blockchain-oriented software engineering: case study in supply chain management." Research and Innovation Forum 2020: Disruptive Technologies in Times of Change. Springer International Publishing, 2021.

[31] Chen, Jiachi, et al. "Defining smart contract defects on ethereum." IEEE Transactions on Software Engineering 48.1 (2020): 327-345.

[32] https://bc24.miage.dev/

[33] Zhang, Rui, Rui Xue, and Ling Liu. "Security and privacy on blockchain." ACM Computing Surveys (CSUR) 52.3 (2019): 1-34.

[34] https://eur-lex.europa.eu/resource.html?uri=cellar:bc4dcea4-9584-11ec-b4e4-01aa75ed71a1.0001.02/DOC_1&format=PDF

[35] https://eur-lex.europa.eu/resource.html?uri=cellar:bc4dcea4-9584-11ec-b4e4-01aa75ed71a1.0001.02/DOC_2&format=PDF

[36] https://cnca-rcrce.ca/wp-content/uploads/2021/05/Executive-Summary-Corporate-Respect-for-Human-Rights-and-the-Environment-Act.pdf