UNIVERSITÉ PARIS 1 PANTHÉON-SORBONNE

DOCTORAL THESIS

Coping with trust issues in collaborative inter-organisational business processes: a blockchain approach

Author:
Eddy KIOMBA KAMBILO

Supervisors:
Pr. Carine SOUVEYET
Dr. Nicolas HERBAUT
Dr. Irina RYCHKOVA

A thesis submitted in fulfillment of the requirements for the degree of Doctor of Philosophy

Université Paris 1 Panthéon-Sorbonne
Information Systems

January 26, 2026

Contents

Chapter 1

General Introduction

1.1 Research Context

In today's digital economy, growing interdependence makes collaboration across organizations inevitable. No single entity can succeed alone. To remain competitive, enterprises must increasingly rely on one another, working together across organizational boundaries to achieve common goals.

Collaboration materializes through inter-organizational business processes (IOBPs), in which enterprises align their operations, share data, and coordinate activities to achieve common goals. One example is a supply chain defined as "the network of organizations involved, through upstream and downstream linkages, in the different processes and activities that produce value in the form of products and services delivered to the ultimate consumer" (Giannakis et al., 2004). In this context, manufacturers depend on suppliers for raw materials, logistics partners for transportation, and distributors to deliver goods to customers. Each partner contributes to the process, and the outcome depends on everyone playing their part.

Trust is central to these collaborations. It is the invisible glue that holds partnerships together, allowing organizations to share information, delegate responsibilities, and rely on each other's performance (Di Ciccio et al., 2019). Yet, building trust remains a challenge. Each organization follow its own processes, rules, and priorities. What one partner values may not be what another expects. Differences in objectives and transparency into each other's operations make trust fragile, uncertain, and often difficult to establish (Akrout, 2015).

Trust can be compromised by many factors. Security vulnerabilities that expose sensitive data, a lack of transparency that generates suspicion, and conflicts of interest may cause partners to act in ways that undermine cooperation, or excessive reliance on intermediaries that concentrate control and risk (Braga et al., 2018). When trust breaks down, collaboration falters. Processes slow, costs rise, and opportunities are lost.

Trust is a complex, multifaceted, and context-dependent concept that is difficult to define. It has been studied across a wide range of disciplines, including organizational science, economics, law, philosophy, and computer science. Philosophy sees it as the decision to rely on others despite uncertainty (Seidl and Mormann, 2014). Law treats it as the foundation of mechanisms such as contracts and liability (Bachmann and Zaheer, 2013). Economics interprets it as a rational calculation of risks and benefits echoed by (Gambetta et al., 2000) who defined trust as "the probability that someone will perform an action that is beneficial or at least not detrimental to us is high enough for us to consider engaging in some form of cooperation with him". In computer science, trust is often operationalized as a measurable or programmable property in systems where autonomous agents must interact without prior knowledge or guarantees (Akrout, 2015). In Organizational science, trust is viewed as a social phenomenon, the willingness of one party to be vulnerable to the actions of another party, based on the expectation that the latter will perform the expected action (Mayer et al., 1995).

Business Process Management (BPM) is used to manage, compose, orchestrate, analyze, and diagnose business processes (Hoonsopon, 2022). BPM has become a key approach for organizations seeking to streamline their operations. Although effective within individual enterprises, BPM faces challenges when extended to inter-organizational contexts. Traditional BPM relies on centralized architectures where a single entity coordinates execution, enforce rules, and maintains data. This configuration creates a *single point of failure*, meaning that the entire process depends on one actor for task execution, coordination, and data protection and Implicit dependencies on trusted intermediaries (Hoonsopon, 2022). If this central entity is compromised, whether through technical failure, operational error, or malicious behavior, the whole process may be disrupted.

This reliance introduces additional vulnerabilities related to scalability, fault tolerance, and security. As a process complexity increases or the number of participants grows, centralized control can become a bottleneck. Furthermore, BPM implicitly assumes a predefined level of trust among participants or the existence of a trusted third party responsible for enforcing rules and ensuring correct execution These limitations underscore the need for alternative approaches or technologies that can enable secure, scalable, and autonomous business process management without relying on centralized architectures (Di Ciccio et al., 2019).

Various technologies and approaches have been proposed developed to address these trust issues: Public Key Infrastructures (PKIs), digital signatures, secure multiparty computation (SMPC), and federated identity systems (Cooper et al., 2008; Lindell, 2020). While these tools enhance aspects of security, authentication, or privacy, they typically rely on central authorities (i.e. institutional trust) or pre-established trust relationships (i.e. social trust), thereby reintroducing dependencies that contradict the goals of decentralization and autonomy. In contrast, Blockchain Technology offers a different approach (Zheng et al., 2018). It promises to embed trust in the decentralized protocol itself rather than in an institution.

Blockchain technology has many features that reinforce trust by design. Its features such as immutability, traceability, process automation, and decentralized execution through smart contracts make blockchain particularly well-suited to enhancing trust in IOBPS (Zheng et al., 2018). Smart contracts (SCs) are self-executing code segments recorded on the blockchain that automatically enforce predefined rules and agreements. Smart contracts ensure that agreed-upon logic is executed automatically and verifiably, without third-party mediation, thereby fostering reliability and reducing the need for external enforcement mechanisms(Bedin et al., 2021). Blockchain creates environments where organizations can collaborate without relying on a single trusted party.

However, blockchain is not a magic solution. It comes with challenges of its own: technical integration with existing information systems, performance and scalability, compliance with legal frameworks, and the cost of implementation or adoption (Zheng et al., 2018). Many Organizations hesitate because of uncertainty about return on investment or a lack of expertise. But despite these challenges, blockchain opens new perspectives on how trust can be managed or even redefined in IOBPs.

This thesis explores trust as a requirement for collaboration in the digital economy. It examines why trust is difficult to establish in inter-organizational processes, and how blockchain might provide new ways to support it. The goal is to understand the potential role and limitations of blockchain as a foundation for trustworthy collaboration between autonomous organizations.

1.2 Research Objectives

To structure and guide this research, we establish a clear research goal along with a set of corresponding Research Objectives (ROs).

> **"The goal of this thesis is to enhance the trustworthiness of collaborative inter-organizational business processes through a blockchain-based approach to trust management."**

We use Wieringa's (Wieringa, 2014) Design Science Research framework to derive the following **Research Objectives (ROs)**:

- RO_1-*Defining Trust:* Define and conceptualize trust within collaborative inter-organizational contexts.
- RO_2-*Analyzing Trust:* We identify and analyze the technical and organizational challenges in addressing trust within enterprise architecture (EA), with a focus on collaborative process design.
- RO_3-*Planning Trust (Design):* Develop an artifact that integrates "trust by design", using blockchain-based features and blockchain design patterns to address the identified trust issues.
- RO_4-*Operationalizing Trust (Design):* Develop a decision-support framework that aligns blockchain design patterns with trust-related requirements and soft goals.
- RO_5-*Operationalizing Trust (Implementation):* Implement a prototype tool to operationalize the framework and facilitate pattern-based design of trustworthy collaborative processes.

RO_1, and RO_2 focus on defining trust and analyzing trust issues in IOBPs: Most Organizations do not start from a blank page: they adapt or redesign their existing processes to improve efficiency, reduce costs, or take advantage of emerging technologies. When collaboration extends across organizational boundaries, the processes must work internally and align with the goals and constraints of external partners (Aalst, 2013). In such situations, trust becomes the key enabler that makes collaboration possible.

A *trust issue* arises whenever one party (the trustor) lacks confidence that another party (the trustee) act as expected. Trust issues can be caused by conflicting interests, lack of transparency, security vulnerabilities, or miscommunication. The stakes become higher when trust extends beyond people to include technological systems. Today, trust has to be established not only between individuals or organizations, but also between humans and technologies, such as artificial intelligence (AI), or distributed ledgers.

The causes and consequences of trust issues are therefore diverse and interconnected. For collaboration to succeed, they must first be made explicit. Defining trust and the conditions under which it deteriorates within enterprise architecture (EA) are therefore essential steps. EA is defined as the fundamental concepts or properties of a system in its environment, embodied in its elements, relationships, and in the principles guiding its design and evolution (ISO/IEC/IEEE 42010). This definition positions EA as a structural and conceptual lens for understanding how organizations coordinate, interact, and govern their processes. Within this perspective, identifying trust issues become the necessary step toward designing collaborative processes that integrate trust by design. Without this foundation, any attempt to improve collaboration risks being built on fragile ground.

RO_3 focuses on planning trust (the integration of trust into process design): We ensure that trust is embedded directly into the design of business processes. We propose the development of an artifact that integrates *"trust by design"*. The idea is to treat trust as a central design principle from the very beginning of process modeling. In practice, Collaborative processes should be conceived and structured in ways that explicitly reduce uncertainty and reinforce confidence between partners.
Blockchain provides an important foundation for this approach. Its intrinsic features such as decentralization, transparency, and immutability make it suitable for situations where independent organizations must cooperate without relying on central authority. These features create a technical baseline that helps foster trust, but they are not sufficient on their own to capture the full complexity of trust-related issues in collaborative environments.

To bridge this gap, we propose an artifact that integrates *blockchain design patterns*. These patterns represent reusable solutions to recurring problems in the design of blockchain-based systems (Xu et al., 2018). By tailoring patterns to the specific trust issues identified in earlier phases, we provide stakeholders with structured guidance on how to embed trust directly into their processes. The use of design patterns is already a well-established practice in software engineering, valued for promoting robustness, efficiency, and security (Six, Herbaut, and Salinesi, 2022b). Applied in the context of blockchain, they can guide practitioners in creating processes that are not only technically sound, but also trustworthy by design.

The RO_4 and RO_5 are operationalized trust: We provide concrete guidance on how to translate trust into technical solutions in collaborative processes. We propose a decision-support framework blockchain features, blockchain design patterns, with trust-related functional requirements and non-functional requirements. As in complex system design, decision-making about which blockchain design to apply in which situations require formal guidance. Without such guidance, it becomes difficult to maintain traceability between the trust issues identified, the requirements they imply, and the technological enablers available to address them.

Our framework formalizes the mapping between blockchain design capabilities, trust requirements, and system-level soft goals through a structured knowledge base and an evaluation model. This reduces the design complexity with traceable justifications for pattern selection, which ensures that trust is embedded not only in intention but also in the final implementation.

In summary, the research objectives outlined above structure the overall trajectory of this thesis: from defining and conceptualizing trust, to embedding it in the design of collaborative processes, and finally operationalizing it through a decision-support framework and prototype tool. To meet these objectives, we develop a set of research contributions presented in the next section.

1.3 Thesis Contributions and Publications

Our research makes several contributions, each constituting a distinct component of the final framework:

[C_1:] Definition of Trust: [P_1]: Building on the seminal works of (Mayer et al., 1995; Mcknight and Clay, 2011; Yan and Holtmanns, 2008; Chopra and Wallace, 2003a; and Pietrzak and Takala, 2021), this phase enabled us to structure the fragmented literature and distinguish three main forms of trust.

Social trust, which governs human relationships both within and outside of organizations shaping cooperation, reputation, and credibility between individual actors (Mayer et al., 1995). *Trust in technology* describes the confidence placed in technical systems, for example, a surgeon to rely on a machine during an operation, or a pilot to trust an autopilot system (Mcknight and Clay, 2011). *Digital trust*, which when human relationships are mediated by technology, such as online transactions or digital platforms, where trust between individuals is reinforced or constrained by technological infrastructures (Pietrzak and Takala, 2021). By distinguishing these three forms, we clarify the multiple dimensions of trust that come into play in inter-organizational settings, and lays the conceptual groundwork for the subsequent phases of this research.

[C_2:] Analysis of Trust issues: [P_1]: We develop a taxonomy of trust issues in Supply Chain Management Systems (SCMS), a domain characterized by multiple stakeholders, data exchanges, and operational interdependence. We then adapt established requirements engineering (documentation, 2018; Pohl (Pohl, 2010)) to translate trust issues into functional and non-functional requirements. Finally, we establish a mapping between trust requirements and blockchain design patterns, thereby providing a structured basis for selecting trust-enabling technologies.

[C_3:] Technology-Aware Enterprise Modeling (TEAEM) [P_2, P_3]: TEAEM extends the Model Driven Architecture (MDA) paradigm. MDA is a software development framework that separates business logic from platform specifics through three primary abstraction layers: Computation Independent Model (CIM) for domain-level requirements, Platform Independent Model (PIM) for system design and functionality, and Platform Specific Model (PSM) for technical implementation details (Mellor, 2004).

TEAEM enhances MDA by introducing a bottom-up constraint propagation and analysis mechanism. We depart from the idea that technical properties defined at the lower MDA abstraction levels can create implicit 'side effects' in the higher MDA levels.

The core objective of TEAEM is to make trust-enabling capabilities explicit and actionable at the conceptual design level of enterprise systems. In the TEAEM, we define trust-related properties at the PSM layer. These properties influence the design of collaborative processes and requirements at the PIM and CIM levels. Specifically, the approach enables the reuse of blockchain features and design patterns as modeling elements within the PSM layer. It supports the tracing of trust issues identified at the CIM layer downward to technical implementations.

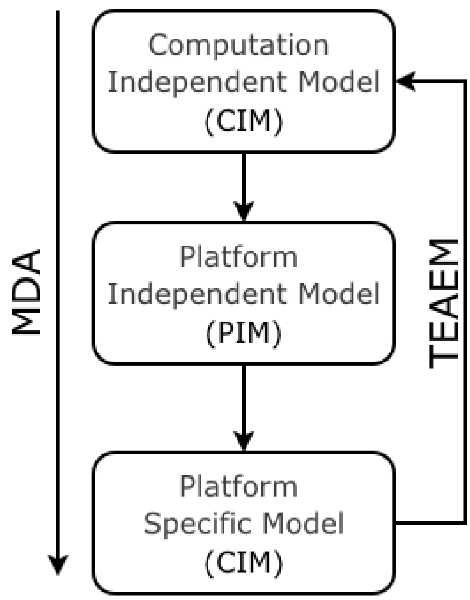

FIGURE 1.1: TEAEM: Extension of the MDA Paradigm

By closing this loop, TEAEM supports design-time validation of trust and align technical enablers (PSM) with high-level collaborative requirements(CIM). In doing so, it strengthens the role of MDA as a methodology not only for structuring enterprise models, but also for embedding trust by design into inter-organizational processes.

[C_4:] Blockchain-enabled Technology-Aware Enterprise Modeling (BC-TEAEM) [P_4] : BC-TEAEM is a decision-support framework that assists system designers in selecting blockchain design patterns based on trust-related requirements and soft goals. This contribution bridges the gap between high-level models and technical solutions. BC-TEAEM is built upon an *ontology* that formalizes the relationships between blockchain patterns and soft goals. This structured knowledge base facilitates the identification of trade-offs among patterns and soft goals.

To make this actionable, we develop a web-based prototype that integrates this ontology with a multi-criteria decision-making method (MDCM). This prototype enables domain users to specify preferences and receive recommendations on the blockchain design patterns best suited to their context.

By combining ontology-based modeling with decision-support techniques, BC-TEAEM enhances traceability in design choices and provides practitioners with structured guidance for embedding trust-enabling technologies. It maps user trust issues into a set of informed, justifiable, and transparent design decisions.

These contributions are published (or in the process of being published) in the following peer-reviewed publications:

- P_1: Kambilo, E. K., Rychkova, I., Herbaut, N., & Souveyet, C. (2023, May). Addressing trust issues in supply-chain management systems through blockchain software patterns. In International Conference on Research Challenges in Information Science (pp. 275-290). Cham: Springer Nature Switzerland.
- P_2: Rychkova, I., Kiomba Kambilo, E., Herbaut, N., Pastor, O., Noel, R., & Souveyet, C. (2024, May). Technology-aware enterprise modeling: challenging the model-driven architecture paradigm. In International Conference on Business Process Modeling, Development and Support (pp. 388-396). Cham: Springer Nature Switzerland.
- P_3: Kiomba Kambilo, E., Herbaut, N., Rychkova, I., & Souveyet, C. (2024, November). Configuration of Software Product Lines Driven by the Soft goals: The TEAEM Approach. In IFIP Working Conference on The Practice of Enterprise Modeling (pp. 262-278). Cham: Springer Nature Switzerland.
- P_4: Kiomba Kambilo, E., Herbaut, N., Rychkova, I., & Souveyet, C. (2025, October). BC-TEAEM: A Decision Support Framework for Blockchain Pattern Selection Based on Soft Goals.
- P_5: Kiomba Kambilo, E., Herbaut, N., Rychkova, I., & Souveyet, C. (Under Revision). Journal PAPER

In parallel, One complementary publications is produced in the broader domain of blockchain technologies, extending the scope of this research and situating its contributions within ongoing discussions in the field.

- P_6: Herbaut, N., Kambilo, E. K., & Ding, Y. (2025, June). A blockchain-based model for fungible assets and secure transformation processes traceability. In International

Conference on Advanced Information Systems Engineering (pp. 121-128). Cham: Springer Nature Switzerland.

P_7: Kambilo, Eddy Kiomba, et al. “A blockchain-based framework for drug traceability: Chaindrugtrac”. Proceedings of the 37th ACM/SIGAPP symposium on applied computing. 2022.

1.4 Thesis Organization

The thesis is structured in two parts. The first part, comprising three chapters, establishes the research background and outlines the methodology guiding this dissertation.

- **Chapter 2**, reviews fundamentals concepts including Trust, Inter-organizational Systems, Business Process Management, Model-Driven Architecture, Enterprise Architecture and Blockchain before reviewing the state of the art on trust within blockchain-based inter-organizational systems. It also examines existing work on trust in blockchain-based collaborative systems and identifies research gaps.
- **Chapter 3** presents the research methodology, based on Design Science Research (DSR), which guides the development, validation, and evaluation of the proposed artifacts throughout this dissertation.

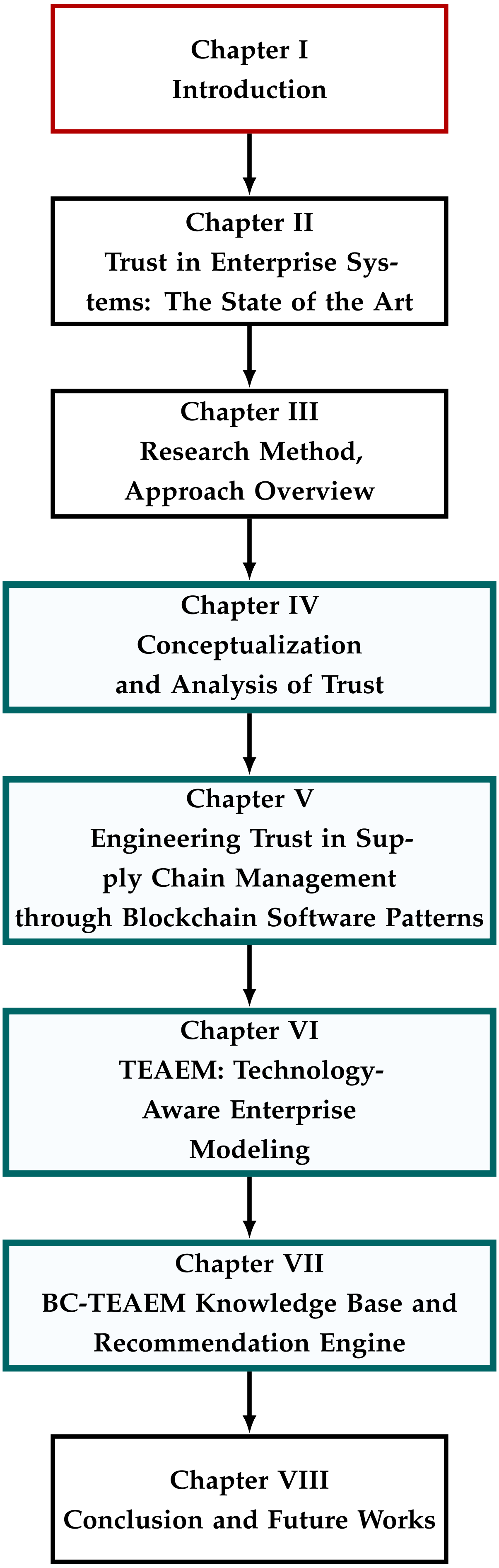

FIGURE 1.2: Structure of the Thesis

The second part, consisting of three chapters, presents the main contributions of this research.

- **Chapter 4** focuses on the conceptualization and analysis of trust issues in inter-organizational systems. It presents a scoping review that identifies and categorizes trust issues from existing literature. The chapter culminates in the development of a taxonomy of trust issues specific to supply chain management systems (SCMS). The chapter also details the methodology for translating trust issues into functional and non-functional requirements, and establishes a mapping between these requirements and blockchain design patterns.
- **Chapter 5** operationalizes the identified trust issues by examining how blockchain features and patterns can be used to address them in the context of supply chain management systems Through a structured pattern-based analysis, it illustrates how design decisions influence trust properties in inter-organizational processes, thereby bridging the gap between conceptual trust analysis and model-driven enterprise design.
- **Chapter 6** presents the Technology-Aware Enterprise Modeling approach, an extension of the Model-Driven Architecture (MDA) paradigm that integrates bottom-up constraint propagation and analysis. The chapter presents the conceptual principles of TEAEM and explains how it incorporates soft goals—representations of qualitative trust-related concerns—into enterprise modeling. It also outlined how TEAEM supports the generation of software artifacts within Software Product Lines (SPL), highlighting its applicability in the design of enterprise systems.
- **Chapter 7** presents BC-TEAEM as a unified contribution. The first part introduces a knowledge base that formalizes the integration of blockchain features and design patterns with soft goals within a structured ontology. The second part details the reasoning and decision-support mechanisms that guide the selection of blockchain patterns according to soft-goal satisfaction built on top of this knowledge base. The chapter concludes with the demonstration and validation of the proposed approach.

Finally, Chapter 8 concludes the thesis and outlines avenues for future research. Each chapter begins with a brief set of bullet points summarizing its objectives and main contributions.

Chapter 2

Trust in Enterprise Systems: The State of the Art

This chapter establishes the theoretical background for architecting trustworthy inter-organizational systems. Its first introduces Enterprise Architecture as a structured framework to align strategic, organizational, and technological issues. It then reviews blockchain technology and its implications for trust in collaborative processes, including its alignment with the Zero Trust paradigm.

The chapter next examines the concept of trust from multidisciplinary perspectives, highlighting its social, technological, and digital dimensions, introducing the notion of *Trust by Design*. Finally, it analyzes existing approaches to engineering trustworthiness formal verification and blockchain-based execution frameworks, and discusses why they remain insufficient for the full lifecycle of IOBPs.

2.1 Enterprise Architecture

Enterprise architecture (EA) is a description of an enterprise from an integrated business and IT perspective intended to improve business and IT alignment, and is used in the majority of large organizations (Kotusev, 2017). EA provides a structural and conceptual for understanding how trust can be embedded into the design of enterprise systems. EA is organized into multiple layers whose alignment ensures coherence between strategic goals and operational execution (McGovern, 2004). *Business levels*, which defines the enterprises' business goals and processes; *Data level*, which specifies the data architecture, including data models, storage and management practices; *Application level*, which describes the software applications and their interactions that support business processes; and *Infrastructure level*, which outlines the underlying technology infrastructure, including hardware, networks, and platforms.

2.1.1 Business Level

The business level specifies the enterprise's business goals, the products and services it produces, what additional products and services its aims to produce in the future, and the constraints that restrict how these goals can be accomplished, including time, budget, legal and regulatory restrictions (McGovern, 2004).

At the Business Level, organizations processes are described through **Business Process (BPs)** defined as sets of activities carried out within a single organization to achieve a specific objective (McGovern, 2004); These processes are managed via **Business Process Management**, which is as a body of methods, techniques, and tools to identify, discover, analyze, redesign, execute, and monitor business processes to optimize their performance (Dumas et al., 2018). While BPM has traditionally focused on internal optimization, the modern landscape is characterized by a shift towards Inter-Organizational Business Processes—–distributed workflows that span multiple independent entities (Alves et al., 2020).

An Inter-Organizational Business Process refers to coordinated sets of activities jointly executed by two or more independent organizations to achieve shared business goals (Bouchbout and Alimazighi, 2011). These processes require data exchange, task distribution, and decision-making across organizational boundaries, supported by shared digital infrastructures and governed by formal or informal agreements.

However, centralized enterprise systems are poorly suited to open IOBPs. Stakeholders are often unwilling to surrender control over their internal workflows to a central authority, yet successful collaboration depends on mutual trust and reliable data exchange (Garcia-Garcia et al., 2020). Consequently, the role of BPM is evolving from mere process control towards enabling trustworthy collaboration. This shift places data privacy, security, and confidentiality—-encapsulated in the concept of digital trust—at the forefront.

New paradigms like blockchain technology respond to this need by embedding features such as trust, transparency, and compliance directly into the technological architecture (Mendling et al., 2018). This positions IOBPs at the intersection of process engineering, governance, and trust management, requiring an interdisciplinary approach to architect reliable collaboration. Blockchain technology, in particular, offers a promising foundation for implementing this **trust by design**.

2.1.2 Data, Application, and Technology Levels

The business vision and processes of an enterprise are operationalized through the data, application, and technology levels of the enterprise architecture. These layers form the backbone of Enterprise systems (e.g., The SCM system in our running example) and directly determine how trust is implemented and managed.

- **Data Level:** defines the integrity, provenance, and confidentiality of the information assets. In this layer, architects define, classify, and manage an organization's data, ensuring it is secure, accessible, and compliant with regulations. They align data management with business goals, support decision-making, and plan for growth or expansion (Spewak and Hill, 1993).
- **Application Level:** governs the logic, interactions, and services that execute business processes. In this layer, architects design, deploy, and integrate enterprise applications, determining their role in business operations, usage policies, and interoperability with existing tools (Spewak and Hill, 1993).
- **Technology Level:** provides the foundational infrastructure (e.g., cloud platforms, blockchain networks) that supports distributed operations. In this layer, architects define and manage the underlying hardware, software, networks, and infrastructure components that support enterprise applications and data. They optimize resources, reduce technical diversity, and ensure systems can scale or integrate to support business goals and new initiatives (Spewak and Hill, 1993).

These architectural design highlight a fundamental insight: trust-related issues or requirements defined at the business level cannot be satisfied in isolation. They propagate across the organizational and technological layers of the enterprise architecture. The choice made at these levels, such as selecting an immutable ledger instead of centralized database impacts the overall system's ability to meet expectations of transparency, integrity, and security.

This interdependence between layers justifies the need to examine the technological solutions capable of operationalizing trust at runtime. Among these technologies, blockchain emerges as a capable solution because it provides features such as decentralization, auditability, and tamper-resistance. Understanding its foundations is therefore essential before analyzing how it can be integrated into enterprise architecture to support trustworthy IOBPs.

2.2 Blockchain Technology

2.2.1 Definition and Fundamentals

Blockchain technology has emerged as a promising solution to enhance digital trust in inter-organisational systems. The first implementation of blockchain technology was proposed in 2008, when Satoshi Nakamoto released a white paper on Bitcoin white paper (Nakamoto, 2008), introducing a decentralized cryptocurrency.

Definition 1 - Blockchain: Blockchain is a decentralized system that allows users to directly interact with each other without the involvement of intermediaries. This eliminates the need for a third party and reduces the chances of fraud or manipulation (Dong et al., 2023).

(Belotti et al., 2019) defines a blockchain as an immutable read-only data structure, where new entries (blocks) get appended onto the end of the ledger by linkage to the previous block's hash identifier.

Both definitions highlight the core characteristics of blockchain: decentralization, immutability, and trust without centralized authorities. This foundation enables advanced features such as automated execution and verifiable interactions.

Data is recorded on a blockchain in the form of authorized transactions that are validated by the network participants.

Definition 2 - Transaction: A transaction in a blockchain refer to the signed data package that stores a message to be sent form an externally owned account. Transactions contain the recipient of the message, a signature identifying the sender, the amount of the ether, and the data to send (Buterin et al., 2014).

After validation, transactions are grouped into the blocks that contains, among other fields, hash of the previous block, a timestamp, and are appended to the chain. Once the block is added, its contents become immutable, any modification would change the block's hash and break the chain (Bedin et al., 2021). To ensure that all nodes maintain a consistent view of the ledger, transactions must be validated and agreed upon by the network. The consensus algorithm plays a critical role in maintaining consistency and trust in this decentralized environment.

Definition 3 - Consensus: A consensus algorithm is a protocol that ensures agreement among distributed network nodes on the current state of the ledger, even in the presence of faulty or malicious nodes (Hussein et al., 2023).

To maintain the same state of ledger for all nodes, the consensus algorithm is applied. Several types of consensus exist, such as Proof of Work (PoW), Proof of Stake (PoS), and Practical Byzantine Fault Tolerance (PBFT). **Proof-of-Work (PoW)** is Bitcoin's consensus algorithm, in which nodes compete to solve a complex mathematical puzzle based on the previous block's hash. The first node to solve the puzzle adds the new block to the chain and receives a reward (Nakamoto, 2008). **Proof of Stake (PoS)** was first implemented in *PPCoin*, introducing the concept of *coin age*, calculated as the product of a coin's value and the duration it is held. PoS grants more rights and rewards to nodes with higher coin age, reducing dependence on computational power (Mingxiao et al., 2017). **Practical Byzantine Fault Tolerance (PBFT)** is a consensus algorithm designed to tolerate Byzantine faults, to addresses transmission errors in distributed systems. PBFT reduces the algorithm's complexity to a polynomial level, significantly improving system efficiency (Mingxiao et al., 2017).

Blockchain technology is not limited to cryptocurrency transactions; It also enables the deployment of decentralized applications through the use of smart contracts.

Definition 4 - Smart Contract: A smart contract is a self-executing piece of code deployed on a blockchain capable of automatically enforcing agreed-upon terms between parties (Di Ciccio et al., 2019). Once deployed, smart contracts operate transparently and irreversibly, minimizing the need for intermediaries.

Blockchain has several intrinsic features that make it relevant for decentralized systems (Viriyasitavat and Martin, 2011).

- *Tamper-proof and Immutability* Once a block is added to the blockchain, its contents cannot be altered without detection. Any attempt to do so would change the block's hash and disrupt its linkage with the subsequent block.
- *Decentralization:* No single entity governs the entire network. Consequently, decentralized applications (DApps) built on smart contracts also inherit this property, as no third party is responsible for executing logic or returning results.
- *Transparency:* All participants can access the content of the blockchain (transactions records, smart contract data, etc.).
- *Traceability:* Every action or transaction can be tracked across the system, enabling accountability and auditability.

- *Integrity:* Ensures that the data stored or transmitted has not been altered or tampered with.

While these features strengthen blockchain's trust model, they also introduce certain limitations that vary depending on the specific use case:

- *Risk of Data Leakage:* Transparency and immutability may expose personal or confidential information. Even if encrypted, such data could become vulnerable over time due to advances in cryptanalysis or the potential compromise of encryption keys (Hughes et al., 2019).
- *Irreversibility of Transactions:* The immutable nature of blockchain means that erroneous or malicious transactions cannot be modified or reversed once validated. This lack of flexibility can create challenges for error correction, regulatory compliance, or dispute resolution (Zheng et al., 2018).
- *Performance Constraints:* Many blockchain networks face scalability issues, including limited transaction throughput and high latency due to the consensus mechanisms and peer-to-peer architecture. These performance bottlenecks can hinder scalability and make blockchain unsuitable for time-sensitive or high-volume applications (Hughes et al., 2019).

In response of these limitations, blockchain design patterns are often used in practice. These patterns encapsulate best practices for addressing known trade-offs, such as improving scalability through off-chain computation, enhancing privacy with zero-knowledge proofs in smart contracts. The use of blockchain patterns is essential to build robust, context-aware solutions that go beyond the default behavior of blockchain platforms.

2.2.2 Blockchain Software Patterns

Pattern-based design has been widely adopted by the software engineering community since the mid-1990s. Software patterns describe recurring designs structures used in software development (Buschmann et al., 1996).

Definition 5: A software pattern is considered as *"a function-form relation that occurs in a context, where the function is described in problem domain terms as a group of unresolved trade-offs or forces, and the form is a structure described in solution domain terms that achieve a good and acceptable equilibrium among those forces"* (Buschmann et al., 1996).

According to (Xu et al., 2018), software patterns play a vital role in addressing trust issues. Blockchain software pattern is a repeatable design solution to a recurring problem in blockchain development. In Dapps, blockchain patterns can provide a systematic way of tackling trust-related issues, such as ensuring data authenticity and integrity, promoting transparency and accountability, and maintaining the privacy and security of the system.

Blockchain patterns have been studied in the literature as reusable solutions to recurring challenges (trust, security, scalability, etc.). Several works have explored how these patterns can address issues in Dapps. To support their organizations and reuse, ontologies have also proposed to formally structure blockchain patterns, such as those (Besancon et al., 2022; Six, Herbaut, and Salinesi, 2022b).

Blockchain's core features combined with blockchain patterns can provide effective solutions for building trust in inter-organisational systems. This combination enables secure and transparent collaboration without centralized control, addressing trust issues.

Blockchain patterns refine how system architects can harness decentralization and immutability to address trust issues. Yet, blockchain remains only one component of the broader enterprise architecture. To fully realize trust by design, blockchain must be integrated with organizational processes, governance structures, and cultural norms. Another major concept influencing modern architectures is the Zero Trust paradigm, which reframes trust not as an inherited property but as a continuously verified one. The following section outlines the principles of Zero Trust and discusses its conceptual alignment with blockchain-based collaboration.

2.2.3 Zero Trust Paradigm

Zero Trust (ZT) is the term of an evolving cybersecurity paradigm that shift protection from fixed, network-based perimeters toward continuous verification of users, assets, and resources (Stafford, 2020). A Zero Trust Architecture (ZTA) applies these principles to the design of Industrial and enterprise infrastructures and workflows. ZT Zero trust assumes there is no implicit trust granted to assets or user accounts based solely on their physical or network location or based on asset ownership (enterprise or personally owned) (Stafford, 2020).

While ZT paradigm means "never trust, always verify", it assumes no actor, devices or network is trusted by default, and access to resources must be continuously verified. Blockchain with it intrinsic features aligns well with ZT principles. It guarantees immutability, decentralization nature that eliminates single points of failure, and cryptographic verification of transactions, but it doesn't automatically enforce ZT principles across Organizations (Aleisa, 2025). This perspective aligns with authors who argue that blockchain is more about enabling control and verifiability than creating trust itself, shifting the focus from trusting parties to trusting the system and its mechanisms.

By implementing a ZT Architecture, blockchain can redefine trust in IOBPs. Whereas traditional IOBPS depend on mix of organizational controls (contracts, laws) and technical platforms, blockchain introduces a paradigm of cryptographic verification. It provides a technical substrates for enforcing “trustless" collaboration. A critical consequence of this model is the potential substitution of technical verifiability for interpersonal trust, suggesting that while blockchain can manage distrust among parties, it may not fully replace the need for trust in human.

To understand the implications of blockchain-enabled ZT architectures for IOBPs, it is essential to recognized that trust is multi-faceted, encompassing not only technical aspects but also social, organizational, and relational dimensions. While blockchain strengthens the technical trustworthiness of systems, social and relational trust—built through reputation, past interactions, and shared norms, remain essential for effective collaboration. This perspective provides the conceptual basis for analyzing trust issues in inter-organizational systems in Chapter 4, where technical mechanisms and social dimensions must continuously interact to achieve common goals.

Taken together, blockchain and Zero Trust highlight a fundamental shift in how organizations approach trust: rather than assuming trustworthy behavior, systems are designed to verify it continuously. However, technical mechanisms alone cannot fully capture the social and organizational dimensions of trust that arise in collaborative settings. To understand how trust is formed, maintained, or undermined in enterprise systems, we next examine its conceptual basis across disciplines.

2.3 Trust in Enterprise Systems

Trust is often understood as a relation between two parties (agent), an agent (the trustor) and another agent (the trustee). The relation is supposed to be grounded on the trustor’s beliefs about the trustee’s capabilities and about the context in which the relation occurs (Taddeo, 2009).

Trust underpins relationships between individuals, businesses, or organizations. Without mutual and positive trust, these relationships cannot thrive. However, trust is a complex and multifaceted concept. Discussing trust involves addressing related aspects such as reputation, risk, vulnerabilities, and other factors that influence cooperation between organizations.

2.3.1 Definitions of Trust

Trust is multidimensional and lacks a single, universally accepted definition. It is perceived differently depending on the context or domain in which it is applied (Chopra and Wallace, 2003b).

In philosophy, trust is often seen a social concept, a willingness to rely on others under conditions of uncertainty. This requires an analysis of intentions, motivations, and mutual expectations (Seidl and Mormann, 2014). The philosopher Niklas Luhmann, views trust as a mechanism for reducing social complexity, allowing individuals and institutions to act without constantly verifying every detail.

In law, trust plays a key role in the formation and execution of contracts, the functioning of legal institutions, and the management of evidence. Legal institutions provide institutional guarantees of trust, through responsibility, transparency, and disclosure obligations. They also a certain degree of reliability in human or organizational interactions. Modern legal frameworks increasingly address trust in digital contexts. (Chopra and Wallace, 2003a) argue that institutional guarantees now intersect with the EU's Digital Services Act (2022), which mandates algorithmic transparency (Art. 13)

In economics, trust reduces uncertainty in exchanges and lowers transaction costs. It lowers transaction costs (Chopra and Wallace, 2003a), facilitates contractual agreements, and encourages investment in long-term partnerships. Economists often distinguish between institutional trust, oriented toward rules, norms, and institutions; and interpersonal trust, which is based on experiences and rational expectations regarding economic partners.

In Information Systems (IS) and computer science, trust plays a key role in the context of service-oriented architectures and decentralized systems, such as blockchain and peer-to-peer networks. Trust is often formalized through scores, reputation metrics, or access control policies (Helo and Ajmal, 2015). This "computable trust" is the subject of dedicated engineering efforts aimed at modeling, quantifying, and in some cases automating trust mechanisms within digital environments.

The table 2.1 is not to provide an exhaustive or systematic review of definitions of trust. Instead, it aims to illustrate the multidisciplinary nature of the concept and show how trust is understood differently depending on the domain in which it is studied or applied.

TABLE 2.1: Key Trust Concepts across Disciplines

Discipline	Author(s)	Key Concept
Philosophy	(Seidl and Mormann, 2014)	Trust as reduction of social complexity
Economics	(Williamson, 1985)	Reduction of transaction costs through trust
Information Systems	(Helo and Ajmal, 2015)	Technical classification of trust in systems
Cross-disciplinary	(Chopra and Wallace, 2003a)	Synthesis of trust concepts across domains

These definitions highlight several key aspects of trust:

- *Relational Nature:* Trust inherently involves a relationship between two parties, where one party (the trustor) places confidence in the actions or intentions of another party (the trustee).
- *Vulnerability:* Trust entails a willingness to be vulnerable to the actions of the trustee, acknowledging that there is a risk involved in relying on another party.
- *Expectation:* Trust is based on the expectation that the trustee will act in a manner that is beneficial or at least not harmful to the trustor.
- *Contextual Factors:* Trust is influenced by the specific context in which the relationship occurs, including the capabilities and characteristics of the trustee.

2.3.2 Conceptualization of Trust

Trust can be conceptualized across multiple dimensions, each capturing different aspects of the trust relationship. Three main form of trust are recognized in the literature : **Social (or interpersonal) trust**, **Trust in technology**, and **Digital trust**.

Social Trust: Social trust reflects (subjective) trustor's beliefs that the trustee has suitable attributes for performing as expected in a specific situation. Social trust constitutes the foundation of any human relationship. (Mayer et al., 1995) defines *trust "as the willingness of one party to be vulnerable to the actions of another party, based on the expectation that the other party will perform the expected action"*. In human relationships, trustor accepts a state of vulnerability, and his dependability from the actions of the trustee. Social trust is often based on psychological or practical criteria. The person can choose to trust not on factual or objective criteria but rather on subjective criteria.

The formation of social trust typically rests on several assumptions (Mayer et al., 1995):

- *Perceived Ability:* The belief that a partner organization possesses the technical skills and competencies required to fulfill its role in process. (i.e., We assume Partner B has the technical capability to securely handle our shared data.)
- *Perceived Integrity:* The belief that the participants will adhere to the agreed-upon rules and protocols when they could benefit from deviating. (i.e., We assume Partner B will not use our shared data for unauthorized purposes).
- *Perceived Benevolence:* The belief that partners will act goodwill and consider mutual interests, not just their own, even when dealing with unforeseen circumstances. (i.e., We assume Partner B will prioritize our shared goals over its own interests).

Trust in Technology: Trust in technology refers to the confidence that users or organizations place in technological systems to perform reliably, securely, and in accordance with expectations. This category of trust encompasses perceptions of system attributes such as functionality, reliability, and helpfulness (Mcknight and Clay, 2011). Trust in technology is particularly relevant in contexts where technology mediates interactions or decisions, such as in information systems, automated processes, or digital platforms.

With the rapid advances of emerging technologies, such as Artificial Intelligence (AI), robotics, and blockchain, organizations increasingly face the challenge of establishing mechanisms to regulate and sustain trust in technology. (Mcknight and Clay, 2011) conceptualize trust in technology as distinct from social trust. While interpersonal or inter-organizational trust facilitates risk-taking and cooperation between actors, it does not fully explain behaviors in technology- mediated contexts, where reliance is placed not on people but on the performance and integrity of systems.

The formation of trust in technology typically rests on several assumptions (Mcknight and Clay, 2011, Meeßen et al., 2020 Viriyasitavat and Martin, 2011):

- *Functionality:* The belief that the system possesses the capabilities required to fulfill its intended purpose (Mcknight and Clay, 2011).
- *Reliability:* refers to the user's belief that the system will operate consistently, correctly, and without unexpected failures (Meeßen et al., 2020).
- *Helpfulness (or Benevolence of Technology):* is the perception that the system is oriented toward supporting the user's goals, providing features that align with user's interests (Mcknight and Clay, 2011).
- *User Satisfaction:* refer to the extent to which a system meets or exceeds the user's perceptions in terms of performance, usability, and overall experience (Viriyasitavat and Martin, 2011).

Digital Trust: Digital trust emerges in IOBPs, where technology plays a role of mediator. Authors in (Ritter, 2020) defines digital trust *"as the confidence users have in the ability of people, technology, and processes to create a secure digital world"*. Digital trust implies that technology can be relied upon to act in accordance with expected standards of integrity, confidentiality, and availability in IOBPs.

This definition underscores that trust in digital environments is multidimensional: it relies not only on the performance of technology but also on the integrity and accountability of the organizations and individuals operating within that environment. In IOBPs, this translates into the expectation that technological systems will uphold integrity, confidentiality, and availability—the core principles of information assurance.

TABLE 2.2: Synthesis of "Trust as..." Concepts

Category	Definition / Description	Key Characteristics	References
Trust as a Social Concept	Interpersonal trust based on a subjective perception from one individual toward another (trustor vs trustee).	• Acceptance of vulnerability • Psychological factors • Subjective criteria (integrity, benevolence, competence)	Mayer et al. (1995), Kari & Chopra (2023)
Trust as a Technological Concept	Trust in a technological system, especially in technology-mediated interactions.	• Reliability • Functionality • Helpfulness • User Satisfaction	McKnight (2011), Meeben (2020)
Trust as a Digital Concept	Digital trust in inter-organizational transactions mediated by technology.	• Based on security, privacy, traceability, • Trust in people, processes, and technologies	Jeffrey (2020), Yan et al. (2008)

Table 2.2 summarizes the different definitions and key characteristics collected from the literature.

In summary, establishing trust in inter-organisational business processes requires a multi-dimensional approach that encompasses a mix of social, technological, and digital trust. Each dimension addresses specific aspects of the trust relationship, and together they form a comprehensive framework for understanding and managing trust in complex, technology-mediated collaborations.

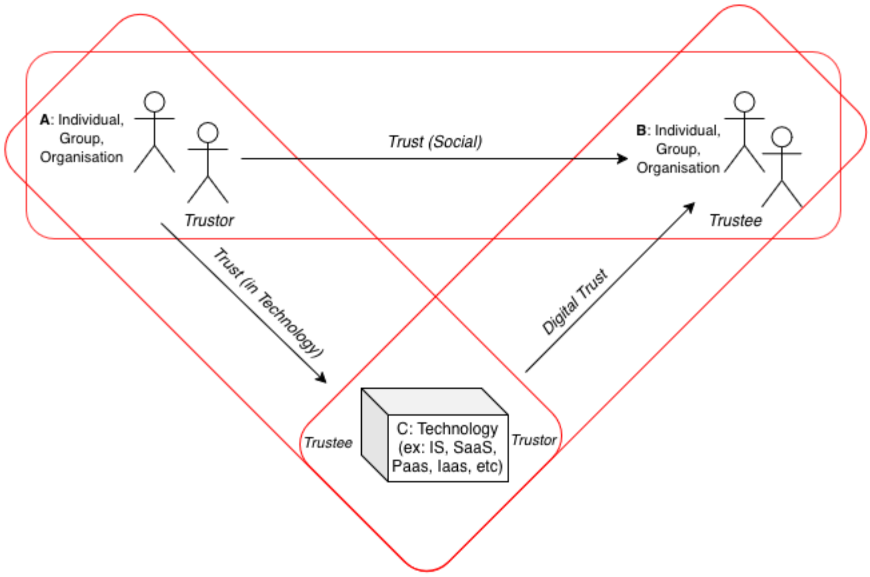

FIGURE 2.1: Categorization of Trust

- Social trust between participating organizations (A-B in Fig. 2.1);
- Trust in technology: to communicate with the other partners or to execute an activity within a process, each party becomes a user of some technological infrastructure and/or information systems and thus needs to establish trust in this technology (A-C in Fig. 2.1);
- Digital trust over the interaction process itself, where technology ensures transparency, reliability, and traceability. (C-B in Fig 2.1).

While the three forms of trust clarify how trust manifests in IOBPs, they do not yet explain how trust can be embedded into system design. Moving from conceptual understanding to engineering practice requires a framework that translates trust expectations into design principles and technical mechanisms. This motivates the notion of *Trust by Design*, introduced in the next subsection.

2.3.3 Trust by Design: The Engineering Approach

The concept of "Trust by Design" refers to the proactive integration of trust- enhancing features and principles into the design and development of systems, processes, or technologies from the outset. This approach aims to create systems that inherently foster trust among users, stakeholders, and participants by addressing trust-related concerns throughout the entire lifecycle of the system.

The three forms of trust t—social trust, trust in technology, and digital trust, explain how trust is perceived and formed in the contexts. Yet none of them fully address how trust can be engineered into enterprise systems. This limitation has led to the emergence of the **"The Trust by Design"** paradigm to embed trust-supporting mechanisms directly within the architecture or design process of enterprise.

Social trust is grounded in expectations about the trustee's ability, integrity, and benevolence (Mayer et al., 1995). Building on Luhmann's view of trust as a mechanism for reducing social complexity through institutionalized expectations and system trust (Luhmann, 2017). Trust by design operationalizes these expectations via explicit structural mechanisms (e.g., transparency, auditability, and rule enforcement) in IOBPs.

Trust in technology derives from perceptions of system functionality, reliability, and helpfulness (Mcknight and Clay, 2011). Building on this view, Trust by Design operationalizes these criteria through verifiable technical guarantees as cryptographic integrity, resilience, and fault-tolerance. Unlike psychological perceptions, these guarantees can be demonstrated through formal models and automated verification procedures, making trust a provable property of the underlying system (Blanchet, 2016).

Digital trust emerges in sociotechnical ecosystems where interactions rely on digital identities, algorithmic decision-making, and platform governance (Janssen et al., 2020). Trust by Design provides the methodological basis for linking technical safeguards (e.g., confidentiality, integrity, accountability) with institutional expectations (e.g., compliance, oversight, auditability). This alignment is increasingly reinforced by regulatory instruments such as the GDPR and NIS2, which require digital infrastructures to provide demonstrable guarantees of trustworthiness (security, transparency, and accountability)

Trust by Design serves as unifying engineering that integrates social, technological and digital dimensions of trust shifting it from an implicit expectation to an explicit design requirement that can be modeled, implemented, and validated within enterprise systems. This perspective is particularly relevant for IOBPs, where stakeholders lack complete mutual trust, and technological mediation plays a central role-—while also providing the conceptual foundation for introducing blockchain as a technology that exemplifies and operationalizes Trust by Design. In offering high-level principles for embedding trust into organizational and technical systems, Trust by Design requires a modeling and architectural methodology capable of expressing trust-related issues across abstraction layers. Model-Driven Architecture (MDA) offers such a framework, and we now examine its role and limitations in engineering trustworthy enterprise systems.

2.4 Architecting Trustworthy Systems

Designing or engineering trustworthy IOBPs requires methodological support capable of modeling, analyzing, verifying, and validating trust-related requirements across organizational and technical layers. Before motivating the choice of model-driven approach for this thesis, it is important to acknowledge that trustworthy systems can be–and–historically have been engineered without relying on emerging technologies such as blockchain.

Two broad traditions dominates this landscape currently: (i) the formal verification methods, which focus on mathematically proving that a system adheres to specified trust properties; and (ii) blockchain-based approaches, which leverage the intrinsic features of blockchain technology to enhance trustworthiness in decentralized systems. Although other approaches exist in the literature, such as, simulation-based analysis, knowledge-based models, ontology-driven approaches, Game-theoretic and Economic models, this section focuses on the two dominant traditions mentioned above because they remain the most mature and influential for verifying distributed, cross-organizational collaborations.

2.4.1 Trustworthiness by Formal Verification

There exist many studies in the literature that demonstrate that trustworthy systems can be constructed through formal verification. **Formal verification** relies on mathematically models and automated solvers to prove the correctness of a system with respect to a formal specification (Clarke et al., 1999). This paradigm has been applied to embedded and distributed systems, cryptographic protocols, and large-scale networked infrastructures.

Many works confirm the central role of formal reasoning in engineering of trustworthy systems. Authors in (Blanchet, 2016) present ProVerif, an automated symbolic verifier designed to analyze security protocols. ProVerif can prove properties such as confidentiality and authentication, which are essential for establishing trust in communication systems. (Kim, 2024) proposes VConMC, an automated verification framework enabling consistency checking for distributed key-value stores, demonstrating that distributed systems can be validated and make trustworthy using formal invariants.

These methods provide rigorous guarantees for trust properties under specific technical assumptions. However, the show some limitations in real IOBPs: they operate at a low abstraction level; they fail to capture high-level business goals and cross-organizational requirements; and they offer no mechanism to align business goals with technical design choices or to trace trust properties back to higher-level organizational models (Clarke et al., 1999).

2.4.2 Blockchain for Trustworthy Systems

Parallel to formal verification, a second dominant approach aims to ensure trustworthy collaboration by relying on blockchain-based frameworks. The tools focusing on generating or enforcing process rules through smart contracts.

Lorikeet (Tran et al., 2018) is illustrative of this paradigm. It provides a model-driven tool capable of transforming business process models into smart contracts, ensuring that process execution adheres to predefined rules on the Ethereum blockchain. Similarly, Caterpillar (López-Pintado et al., 2019) provides a business process execution engine on Ethereum, enabling correct execution and tamper-proof auditability of inter-organisational workflows.

These tools operationalize trust by embedding process logic directly into blockchain smart contracts. Nevertheless, their focus remains to the execution layer, following the top-down design process. They lack mechanisms to analyze how architectural choices impact trust-related qualities at the high-level organizational models. They also lack capacity to reason about trade-offs between different trust dimensions (e.g., privacy vs transparency) during the design phase, and offer no mechanisms to propagate architectural decisions upward to strategic or organizational models. Thus, while valuable for trustworthy execution, these tools do not support the full architectural reasoning required in inter-organizational systems.

2.4.3 Limitations of Existing Approaches

Both formal verification and blockchain-based approaches provide valuable tools for engineering trustworthy systems. However, they exhibit limitations when applied to the full lifecycle of inter-organizational business processes. Formal verification operates at a low abstraction level, failing to capture high-level business goals and cross-organizational requirements. Blockchain-based tools focus on execution, lacking mechanisms to analyze how architectural choices impact trust-related qualities at higher organizational levels. Neither approach offers mechanisms to align business goals with technical design choices or to trace trust properties back to higher-level organizational models. These limitations motivate the need for a model-driven methodology capable of bridging organizational and technical layers while embedding trust considerations throughout the design process.

2.4.4 Bridging the Gap: The Case for a Model-Driven Methodology

Despite the reason evoked above (2.4.3), IOBPs introduce challenges that neither formal verification nor blockchain-based approaches fully address. In IOBPs, trust issues and requirements emerge in the same time at multiple layers of the EA: organizational policies, business processes, information systems, and technological infrastructures. Modelling such systems therefore requires a multi-level abstraction, traceability (top et bottom-up), and alignment mechanisms that formal verification and blockchain-based tools alone do not provide.

The Model-Driven Architecture (MDA) provides a fundamental framework for the design and development of enterprise information systems, prioritizing the top-down design process that separates business logic from technological implementation. MDA aligns business goals (CIM layer: represents the system's requirements and business context without detailing the structure or processing;), system design functionality (PIM layer: specifies the system's structure and functionality but abstracts away the details of any specific implementation platform), and technological implementation (PSM layer that provides the technical details on system implementation using a particular technology or platform.) (Mellor, 2004).

This separation of issues is particularly valuable across organizational boundaries, where partners must collaboratively articulate requirements at the business level before deciding how such requirements should be instantiated technologically. Unlike formal verification and blockchain-based approaches, MDA offers a holistic framework capable of capturing trust-related issues across abstraction layers while ensuring traceability and alignment between organizational goals and technical design choices.

Furthermore, IOBPs involve a high degree of technological heterogeneity, with partners employing diverse platforms, protocols, and standards. By enabling platform-independent modeling, MDA facilitates interoperability and integration across disparate systems. Also, it prevents premature technological choices from constraining architectural analysis and ensures trust-related requirements can be evaluated independently of specific platforms. This is crucial when technologies such as blockchain have a side effect on organizational strategy and processes. Recent studies highlight how architectural decisions—especially regarding distributed ledgers—shape organizational strategy and may introduce rigidity or unexpected trade-offs if not analyzed at design time (Morkunas et al., 2019; Rajnak and Puschmann, 2021).

For these reasons, MDA constitutes an appropriate methodological foundation for modelling trust in inter-organizational business processes. It supports multi-layer, enabling alignment between business goals and technical design choices, and provides the representational structure necessary for embedding trust issues or requirements into system design. Nevertheless, classical MDA remains insufficient for capturing multi-dimensional notions of trust, managing trade-offs between trust attributes, or analyzing bottom-up impacts of technical choices (e.g., consensus algorithms, data) on business-level goals. These limitations motivate the need for an extended, technology-aware modelling approach.

2.4.5 Towards Trust-Driven Model-Driven Architecture

The above analysis reveals a methodological gap. Existing trustworthy-system methods either operate at too low a level (formal verification), or focus narrowly on execution (blockchain frameworks), or provide architectural alignment without explicit trust modelling (classical MDA). None of these approaches fully address the requirements of Trust by Design in inter-organizational contexts, where trust must be represented as a multi-dimensional, cross-cutting concern.

This thesis therefore proposes an extended model-driven methodology, TEAEM, which integrates trust modelling into architectural analysis and incorporates technology awareness into MDA. TEAEM enables both top-down propagation of trust requirements and bottom-up evaluation of technological impacts, thereby providing a unified foundation to engineer trustworthy inter-organizational systems.

2.5 Conclusion

This Chapter establishes the multidisciplinary foundation for addressing the core research problem: architecting for trust in blockchain-based inter-organizational systems. We start by defining the concepts such as Enterprise Architecture and its layered structure, framing how trust can be embedded into system design. We discuss also on IOBPS that highlight the critical demand for digital trust in cross-boundary workflows.

Then, we review the blockchain technology, its foundational principles such as decentralization and immutability, its enabling feature—-the smart contract, and its inherent limitations. This analysis underscores the necessity of blockchain patterns to address design trade-offs and positions blockchain as a technical substrate of "trustless" collaboration aligned with Zero-Trust principles. The review of trust notions reveals it tripartite nature, social, technological, and digital, demonstrating that trust in IOBPs is a complex interplay between confidence in partners, reliance on technology, and assurance in mediating processes.

We identify some limitations in the current MDA approaches: while for functional alignment, they lack the mechanisms to model and trace the long-term strategic impact of architectural choices (i.e., bottom-up traceability), especially those involving technologies such as blockchain, on non-functional qualities such as trustworthiness. This limitation manifests in three core challenges:

- *Conceptual Gap:* the absence of a unified, operationalizable model to integrate the multiple dimensions of trust into the design process.
- *The Methodological Gap:* the inability of existing approaches to account the side effects of technological choice in the low levels (PSM) on the higher-levels (PIM-CIM).
- *The Practical Gap:* The lack of guided methodologies for practitioners to navigate trade-offs between competing trust issues (e.g., transparency vs. confidentiality) in blockchain-based IOBPs.

This synthesis, illustrated through a pharmaceutical supply chain example, clarifies the research imperative. A novel approach or framework is necessary to bridge this gap. It must facilitate the explicit modeling, analysis, and traceability of trust issues from business strategy to technological instantiation, enabling architects to make informed decisions that sustain trustworthiness over the system's lifecycle.

Chapter 3

Research Method, Approach Overview

The chapter adopts the Design Science Research (DSR) methodology, which provides a rigorous process for addressing complex socio-technical problems through the creation and evaluation of artifacts.

We introduce this framework as a structured response to trust-related issues in IOBPs. The framework integrates blockchain features, patterns and model-driven engineering to support modeling, analysis, and operationalizing trust through three core objectives: *defining, planning, and operationalizing trust*.

To anchor the discussion and support progressive illustration of this thesis contributions, we introduce a running example drawn from the Supply Chain Management (SCM) domain. This example is used to illustrate Chapters (4, 6 and 7).

This chapter is organized as follows:

- Section 3.1 presents introduces the Design Science Research framework and the research questions that guide this thesis.
- Section 3.2 presents the developed artifact in the context of DSR.
- Section 3.3 describes the running example used to illustrate the framework throughout the thesis.
- Section 3.4 concludes the chapter.

3.1 Research Method

3.1.1 Design Science Research Methodology

Design Science Research (DSR) is a research methodology that focuses on the design and evaluation of artifacts to solve practical problems. In Information Systems (IS), DSR is distinguished from behavioral science by its emphasis on creating solutions (methods, models, processes, algorithms, or tools) that extend human and organizational capabilities (Hevner et al., 2004).

Design science is the design and investigation of artifacts in context. The artifacts we study are designed to interact with a problem context in order to improve something in that context. According to (Wieringa, 2014), DSR in Information Systems involves two core activities: designing an artifact that improves something for stakeholders, and empirically investigating a performance of the artifact in a specific context. The designed artifact can be methods, models, techniques, algorithms, constructs used in Information Systems.

In this Thesis, we adopt the (Wieringa, 2014) DSR approach. The Figure 3.1 illustrates the DSR process as described by Wieringa and adapted to this thesis structure.

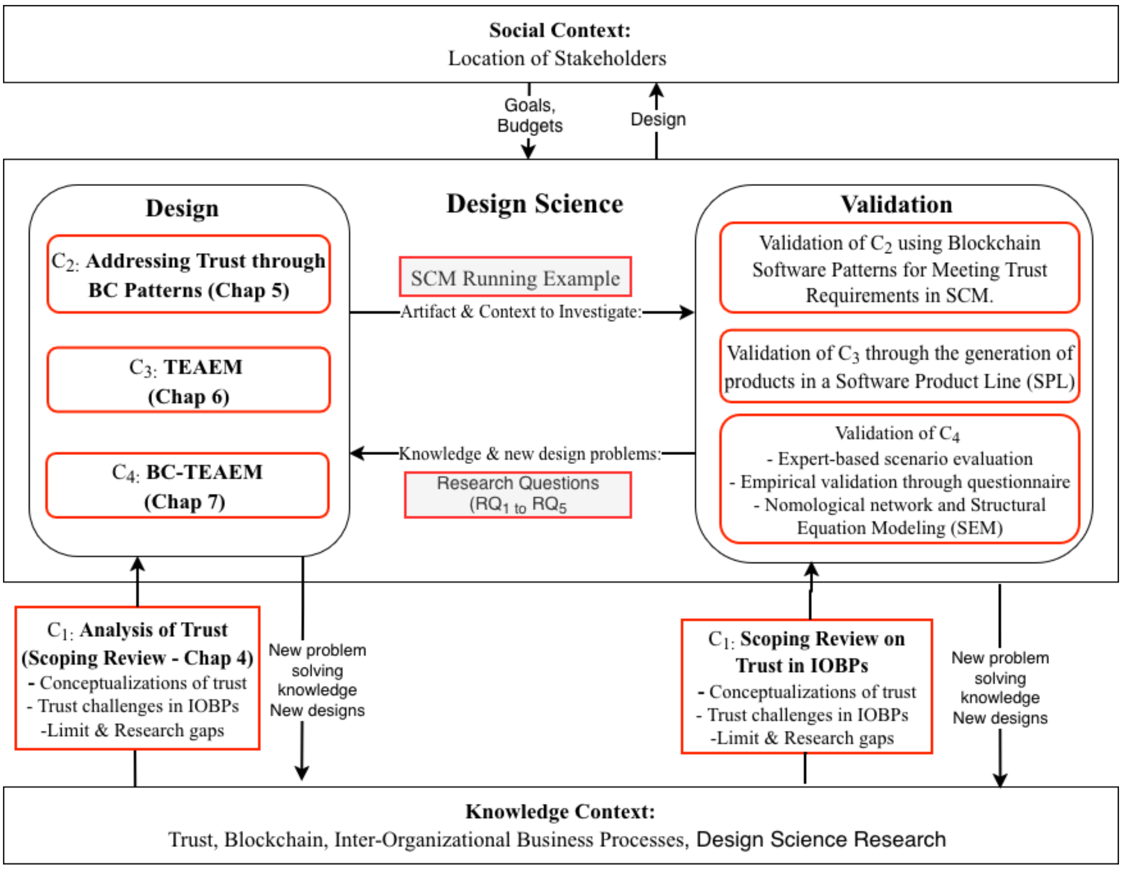

FIGURE 3.1: Adapted Design Science Research by Wieringa

The **social context** contains stakeholders who may affect the project or may be affected by it. Stakeholders include users, operators, maintainers, instructors, etc., of the artifact to be designed. Sponsors provide the budget for the project and set goals to be achieved by the project (Wieringa, 2014).

The **Knowledge context** consists of existing theories from science and engineering, specifications of currently known designs, useful facts about currently available products, lessons learned from the experience of researchers in earlier design science projects, and plain common sense.

The **Design Science Research Cycle** follows the framework illustrated in Figure 3.1 and consists of multiple interconnected cycles:

1. **Problem investigation**: Consists of identifying the problem context: including stakeholders, goals, the conceptual problem framework, involved phenomena, their causes and consequences, as well the knowledge context and existing gaps.

   - **C_1: Scoping Review on Trust in IOBPs (Chapter 4)**: The initial problem investigation addresses the research questions by examining conceptualizations of trust, identifying trust challenges in IOBPs, and establishing limitations and research gaps.
   - **C_{1b}: Analysis of Trust (Chapter 4)**: This cycle further investigates the artifact and context, focusing on trust requirements in Supply Chain Management. A running SCM example grounds this analysis.

2. **Treatment design**: Consists of designing one or more artifacts that could treat the problem identified. Requirements contribute to Goals? Available treatments.

   - **C_2: Addressing Trust through Blockchain Patterns (Chapter 5)**: This cycle focuses on using blockchain software patterns to address identified trust issues in supply chain management systems. It establishes a mapping between trust requirements and blockchain patterns from a Trust by Design perspective.
   - **C_3: TEAEM Approach (Chapter 6)**: This cycle focuses on designing the TEAEM approach, which integrates trust requirements into enterprise architecture modeling. It extends Model-Driven Architecture with trust-oriented modeling and bottom-up constraint propagation.
   - **C_4: BC-TEAEM Framework (Chapter 7)**: This cycle extends the TEAEM design with blockchain capabilities and produces BC-TEAEM framework, which operationalizes trust through a blockchain ontology and decision-support system. It includes multi-criteria evaluation and pattern recommendations.

3. **Treatment validation**: The produced artifacts are evaluated to assess whether they treat (satisfy) the problem. Each cycle incorporates validation activities that generate new knowledge and identify new design problems:

   - **Validation of C_2**: Through use case scenarios of pharmaceutical supply chains process by using Blockchain Software Patterns and features to address trust issues.
   - **Validation of C_3**: Implemented through the generation of products in a Software Product Line (SPL).
   - **Validation of C_4**: Conducted through expert-based scenario evaluation, empirical validation via questionnaire, and analysis using nomological networks and Structural Equation Modeling (SEM).

3.1.2 Research Questions

The goals of researchers are to contribute to the body of knowledge in a specific field, while the goals of external stakeholders are to solve a problem or improve a situation in a specific context (Wieringa, 2014).

In this thesis, we focus on the goals of external stakeholders, which are introduced in Chapter 1. To address these goals, we formulate five research questions $RQ_{1–5}$ each aligned with a stage of the DSR cycle and mapped to the corresponding chapters.

- RQ_1: How can trust issues defined and conceptualized in collaborative inter-organizational business processes (Problem identification - Chapter 4)
- RQ_2: How can blockchain technology contribute to addressing trust issues in these processes? (Problem identification - Chapter 4 and Chapter 5)
- RQ_3: How can an artifact be designed to embed "trust by design" principles to address identified trust issues in collaborative processes? (Treatment design - Chapter 6)
- RQ_4: How can a decision-support framework be developed to align design solutions with trust-related requirements and non-requirements? (Treatment validation - Chapter 7)
- RQ_5: How can this framework be operationalized through a tool that supports the design of trustworthy collaborative processes? (Treatment validation - Chapter 7)

The research questions highlight the problem space and the knowledge gaps to this thesis aims to address, leading us to the development of the proposed artifact. The Table 3.1 summarizes how these research questions map to the chapters and artifact components.

TABLE 3.1: Research Questions and Corresponding Chapters

Research Question	Addressed in	Artifact Component Used
RQ_1	Chapter 4	Taxonomy of trust issues
RQ_2	Chapter 4, 5	Trust issues analysis
RQ_3	Chapter 6	TEAM
RQ_4	Chapter 7	BC-TEAEM ontology
RQ_5	Chapter 7	BC-TEAEM Recommendation engine

3.2 Proposed Artifact Overview

Our proposed artifact is the **Blockchain Technology-Aware Enterprise Architecture Modeling (BC-TEAEM)** framework. He is designed to address this need by integrating blockchain technology with the Technology-Aware Enterprise Architecture Modeling approach. The framework is structured around three main objectives: defining trust, planning trust, and operationalizing trust.

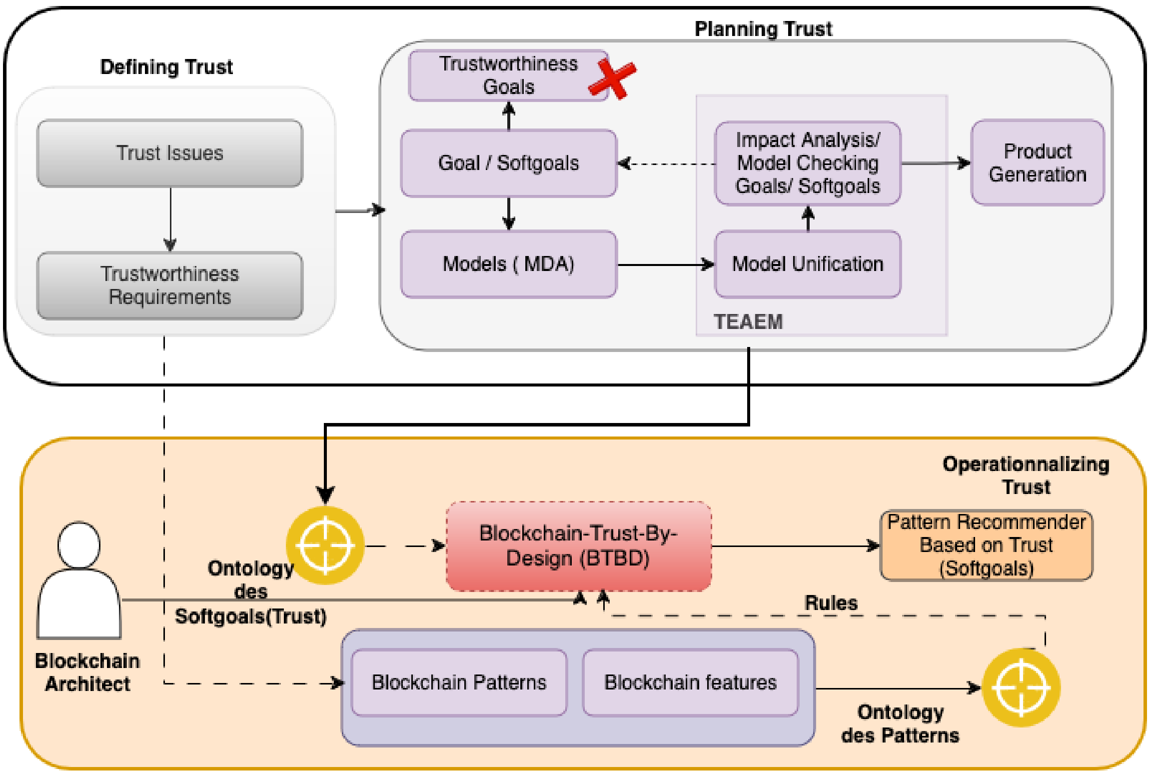

FIGURE 3.2: Artifact Global Overview

As show in Figure 3.2, the framework is composed of two main components that work together to achieve these objectives:

1. **TEAEM**: a modeling and analysis approach extending Model-Driven Architecture (MDA) with trust-related soft goals and bottom-up constraint propagation;

2. **BC-TEAEM**: a reasoning-enabled decision-support system integrating a blockchain ontology, multi-criteria evaluation, and pattern recommendations.

TEAEM Component: The TEAEM supports the planning of trust in collaborative processes. It is designed to bridge the gap between business goals and technical choices in IOBPs by integrating trust issues and requirements into the modeling phase. It extends the Model-Driven Architecture–which separates business logic from technical implementation through model-to-code transformations—with a trust-oriented modeling framework. This addresses a key weakness in MDA: the frequent inconsistencies between business objectives and technical decisions, which can lead to misaligned trust requirements, financial losses, and eroded partnerships. To overcome this, TEAEM introduces a bottom-up constraint analysis that systematically detects and refines these inconsistencies, ensuring that business goals and technical choices are aligned with trust requirements before any code or product generation. The tools enabling this iterative, trust-driven process are detailed in Chapter 6.

BC-TEAEM Component: BC-TEAEM operationalizes trust through a domain-specific ontology that formalizes relationships between blockchain features, design patterns, and soft goals. This ontology serves as a knowledge base for automated reasoning, allowing the system to detect potential conflicts between patterns and soft goals (trust dimensions), thus providing a transparent and justifiable basis for architectural decisions. Additionally, the framework incorporates a dynamic Multi-Criteria Decision-Making (MCDM) module. This module processes user-defined preferences to generate ranked pattern recommendations, enabling structured trade-off analysis. Decision-makers can explore alternative design choices and assess their implications on trust through an interactive user interface that visualizes reasoning outcomes and recommendations. The complete BC-TEAEM tool, including its ontology, knowledge base, and interface is detailed in Chapter 7.

3.3 Running Example

Our example addresses the problem of traceability in pharmaceutical supply chains. The medical supply chain handles life-critical products such as vaccines, surgical implants, and prescription drugs. In this context, any breach of trust, including counterfeit products, improper storage conditions, or missing provenance information, can have serious fatal consequences.

Drawing on the example from the literature (Di Ciccio et al., 2018), we consider a network of actors involved in delivering a high-value, temperature-sensitive cancer medication from its manufacturer to a hospital pharmacy. The process begins at the manufacturing site, where the medication is produced under strict regulatory insight. The products are then transferred to a logistics provider responsible for preserving cold-chain conditions throughout transport. Along the way, the shipment passes through several points of interaction, including regional warehouses, customs checkpoints, and hospital receiving area, where different stakeholders handle and monitor the drugs. The process concludes at the hospital pharmacy, which dispenses the medication to the oncology department for patient treatment.

In such a chain, traceability plays a central role. The hospital must be able to verify that the medication its receives is authentic, has been stored within the required temperature range, and originates from an authorized manufacturer. The logistics provider must be able to demonstrate that cold-chain conditions were continuously maintained during transportation. Regulatory bodies must have confidence in the completeness and accuracy of the audit trail to certify compliance. And in the event of a product recall, every participant must be able to identify, quickly and precisely, which batches are affected and where they are located.

Several challenges emerge at different points along the supply chain:

- The hospital worries about counterfeit products entering the chain through intermediaries.
- The logistics provider fears disputes over cold chain failures if temperature readings are challenged.
- Regulators need auditable and tamper-proof records for certification and post-market surveillance.
- The manufacturer wants to ensure that sensitive production data remains confidential while still providing necessary provenance information.

3.4 Conclusion

This chapter introduces the methodological foundation that guide the research presented in this thesis. The DSR methodology provides a structured framework that links problem investigation, artifact design, and artifact validation. By adopting this approach, the research ensures rigor and relevance, addressing both theoretical contributions and practical applications.

The five research questions (RQ_1–RQ_5) formulated within this framework structure the contributions developed in the subsequent chapters. These questions span from identifying trust challenges in inter-organizational business processes to designing and validating blockchain-enhanced artifacts for trust management in supply chain contexts. Each research question corresponds to specific cycles of the DSR process, ensuring a coherent progression from problem understanding to solution development and evaluation.

As illustrated in the research design, this work progresses through multiple interconnected cycles, beginning with a scoping review to establish the knowledge base and research gaps, followed by the design of trust-enhancing artifacts, and culminating in comprehensive validation through multiple methods. This methodological approach not only supports the development of novel contributions but also facilitates the generation of actionable insights for both academia and practice.

The following chapters implement this methodology, detailing each cycle of the research process and demonstrating how the DSR framework enables the systematic creation and evaluation of artifacts that address challenges in blockchain-enabled supply chain management.

Chapter 4

Conceptualization and Analysis of Trust in SCM

The purpose of this chapter is to establish the theoretical foundation required to understand trust issues inherent in inter-organizational business processes. The chapter proceeds in two main parts. First, drawing from multidisciplinary perspectives, it synthesizes how trust is defined and conceptualized in Supply Chain management domains. Second, its adopts a structured scoping review methodology to identify, classify, and interpret issues reported in the literature, and to examine how blockchain technologies and blockchain software patterns can address them. This chapter focuses on the conceptual and methodological definition of a mapping approach between trust-related concerns and blockchain patterns, supported by an illustrative running example.

4.1 Scoping Review Methodology

This chapter adopts the scoping review methodology (Arksey and O'malley, 2005) to systematically explore how trust is defined, conceptualized and addressed in the literature on inter-organizational business processes. A scoping review is particularly appropriate for this research because the notion of trust spans multiple disciplines, lacks unified conceptualization, and is characterized by heterogeneous terminologies. Rather than aiming at theory testing or effect measurement, the objective is to map existing knowledge, identify dominant conceptual patterns, and reveal gaps that motivate further conceptual and design-oriented research.

The results of the scoping review constitute the first contribution of this thesis (C_1), which offers a taxonomy of trust issues and a mapping between these issues and blockchain features and software patterns.

4.1.1 Research Questions

We answer to the first two research questions of the thesis:

- **RQ_1:** How can trust defined and conceptualized within inter-organizational business processes?
- **RQ_2:** How can blockchain technology contribute to addressing trust-related issues in such processes?

4.1.2 Search Strategy

Given the interdisciplinary nature of trust, we design the search strategy to ensure boarder coverage across multiple academic domains, including management, information systems, organization theory, blockchain, supply chain management, and information technology. Searches were conducted in the following electronic databases: *Scopus, SpringerLink, IEEE Xplore, ScienceDirect, ACM Digital Library, and Google Scholar*.

Search queries combined terms related to trust (e.g., trust, trustworthiness, digital trust), inter-organizational contexts (e.g., inter-organizational processes, collaborative processes, supply chain), and enabling technologies (e.g., blockchain, distributed ledger, digital platforms). The search was not restricted to a single domain in order to capture conceptual diversity.

To mitigate limitations related to database indexing and keyword variability, we complement the databases search with *backward snowballing:* screening reference lists of key papers, *forward citation searching:* identifying studies citing on seminal works, and *manual searches:* in specific journals and conference proceedings.

4.1.3 Study Selection

The selection process was conducted in two phases: (1) an initial screening based on titles and abstracts, and (2) a full-text assessment of the remaining studies. At each step, decisions were guided by relevance to inter-organizational contexts, conceptual engagement with trust, and consideration of technological or organizational mechanisms used to address trust challenges.

Inclusion Criteria: During the first screening phase, studies were included in the review if they satisfy the following criteria:

- (I_1) They explicitly or implicitly discussed issues related to trust (social trust, digital trust, institutional trust, or trust in technology) within inter-organizational contexts, particularly in supply chain management.
- (I_2) They provide a conceptual, analytical, or empirical mechanisms, approaches, or frameworks aimed at addressing trust-related issues or improving trust within supply chain management.
- (I_3) They examined, proposed, or discussed the use of blockchain as a potential enabler or mediator for addressing trust issues in SCMS

Exclusion Criteria: During full-text screening, studies were excluded when they met at least one of the following criteria:

- (E_1) They focused exclusively on intra-organizational processes, without considering interactions among independent organizations, or external stakeholders.
- (E_2) They presented purely technical or engineering-oriented blockchain implementations that lacked consideration of trust-related concepts or social dimensions.
- (E_3) They lacked sufficient conceptual or theoretical depth, offering limited analytical insight into the nature, dimensions, or dynamics of trust.

Refining Ambiguous Cases: During the screening process, some studies presented ambiguous terminology, for example, using "trust," "reliability," or "confidence" interchangeably. To address this, we followed (Arksey and O'malley, 2005) recommendation of applying interpretive judgment while ensuring consistency:

When ambiguity arose, we discussed within the research team (as recommended in scoping review methodology) to assess whether the authors addressed trust as:

- **relational construct** (e.g., vulnerability, expectation, credibility), or
- **technical property** (e.g., uptime, reliability of a system)

Only studies engaging with the relational or socio-technical dimensions of trust were retained. This interpretive step was necessary due to the inconsistent use of terminology across disciplines.

4.1.4 Data Extraction and Charting

Data extraction was conducted using a structured coding scheme aligned with the research questions:
For $\mathbf{RQ}_1$, the extraction focused on:

- explicit definitions of trust,
- conceptual dimensions, forms, and levels of trust,
- underlying assumptions and conceptual boundaries.

For $\mathbf{RQ}_2$, the extraction targeted:

- trust-related issues identified in SCMS,
- technological mechanisms proposed to address these issues,
- the role attributed to blockchain and related technologies.

The extracted data were organized through a thematic synthesis approach, allowing patterns to emerge across domains while preserving contextual nuances.

4.2 Results: Data Collection and Analysis

This section presents the results of the scoping review based on the corpus of selected studies. The objective is to report how trust and trust-related issues are addressed in the literature on inter-organizational business processes, particularly when blockchain technologies are involved in accordance with the data extraction strategy defined in the previous section. The results are organized around the two research questions, without interpretative or evaluative claims.

4.2.1 Overview of the Reviewed Literature

The selected studies span multiple academic domains, including supply chain management, information systems, organizational studies, healthcare, finance, and digital platforms. The corpus reflects the interdisciplinary nature of trust research, with contributions ranging from conceptual and theoretical works to applied studies examining technology-mediated collaboration.

4.2.2 Results for RQ_1: Definition of Trust in the Literature

This subsection reports how trust is defined and conceptualized across the reviewed studies.

Definition of Trust across different domains

The reviewed studies indicate that trust does not have a single, universal meaning and that its definition varies across disciplines and application domains. For example, in supply chain management, trust is often associated with predictability and the quality of information shared by partners (J. Wang et al., 2022). In digital systems, trust increasingly shifts toward the perceived reliability, security, and robustness of the technological infrastructures (J. Wang et al., 2022).

This diversity of perspectives highlights that trust cannot be analyzed in uniformly across contexts. Depending on whether the focus is social, organizational, or technological, different challenges emerge. Clarifying how trust is defined within inter-organizational processes is therefore a prerequisite for any systematic analysis or design-oriented.

Table 4.1 provides an overview of representative trust definitions across multiple domains relevant to inter-organizational collaboration.

TABLE 4.1: Different definitions of trust across various domains

Trust Definition	Study	Trust Definition Example
Positive Expectation	[Mayer et al., 1995], [Patrick et al., 2005]	The willingness of one party (trustor) to be vulnerable to the actions of another party (trustee), based on the expectation that the other party will perform the expected action.
Social Expectation	[Mattila, Seppälä, et al., 2016]	a social interaction where one party's subjective probability about whether the other party will perform a particular action that benefits the the trustor.
Individual Perception	[Chopra and Wallace, 2003b]	is an attitude held by an individual. It is influenced by the personality of the trustor and by the attributes of the recipient, and it in turn influences the behaviour of the trustor, but it is equivalent to none of them.
Belief in Conduct	[Y. D. Wang and Emurian, 2005]	An actor's confidence that another will manage resources, achieve goals, or execute tasks without misuse.
Trustworthiness	[Gräbner et al., 2021]	refers to the confidence that organizations place in another's reliability and integrity.
System-based Trust	[Mühlböck, 2022]	Trust grounded in institutional structures, rules, and technological systems rather than interpersonal familiarity.
Algorithmic Trust	[Morley, 2020], [Mittelstadt, 2022]	Confidence placed in automated systems to act reliably and ethically in decision-making processes, based on transparency, accountability, and explainability.
Digital Trust	[Voloch et al., 2019]	refers to the confidence that individuals or organizations have in the reliability, security, and ethical use of technology as a mediator in interactions, transactions, or data exchange.

Despite their diversity, these definitions share common recurring elements:

- Trust involves an acceptance of vulnerability.
- Trust reflects expectations about how partners will behave.
- Trust functions as a mechanism to reduce uncertainty in collaborative environments.

Presenting trust in this way clarifies why a single definition is insufficient for complex, technology-mediated environments. This observation also justifies the need to distinguish between social trust, trust in technology, and digital trust in the remainder of this thesis.

Forms of Trust in Inter-Organizational Relationships

The diverse definitions of trust synthesized on the previous sub-section 4.2.2 reveal that trust is not a monolithic concept but a multifaceted construct. To answer this sub-question, we consolidate these perspectives into different categories that captures the multifaceted nature of trust that emerge in inter-organisational contexts.

The literature identifies several types of trust, reflecting whether it is grounded in social relationships, institutional arrangements, technological systems, or digital interactions. Common classifications include *Interpersonal Trust*, *Organizational Trust*, *Institutional Trust*, *Technological or System-based Trust*, and *Digital Trust* (Mayer et al., 1995; Mcknight and Clay, 2011; Ritter, 2020).

Interpersonal Trust refers to confidence between individuals and is often grounded in perceived integrity, competence, and benevolence (Lauster et al., 2020). Interpersonal trust is critical in collaborative networks where long-term relationships support information sharing and coordination. *Organizational Trust* extends this notion of confidence in organizational policies, processes; and decision-making practices (Rusu and Baboş, 2015). *Institutional Trust* relies on formal rules, regulations, or industry standards to reduce uncertainty (Rusu and Baboş, 2015).

Technological or System-based Trust emerges when systems or platforms mediate interactions between stakeholders. In these cases, trust is placed in the correct functioning of systems rather than in specific human actors (Mcknight and Clay, 2011). *Digital Trust* further extends this perspective by emphasizing transparency, traceability, accountability, and data integrity in digitally mediated interactions (Ritter, 2020)

For analytical clarity and to align with the socio-technical nature of modern Inter-Organizational Business Processes (IOBPs), we synthesize these perspectives into three types, interrelated categories: `Social Trust`, `Trust in Technology`, and `Digital Trust`. This typology provides a foundation for analyzing trust issues in IOBPs.

Social Trust: reflects (subjective) trustor's beliefs that the trustee has suitable attributes for performing as expected in a specific situation. This category encompasses the interpersonal and organizational forms of trust that arise from social interactions between individuals or organizations. It involves beliefs about the integrity, benevolence, and competence of others (Mayer et al., 1995), as well as the willingness to be vulnerable based on these beliefs. Social trust is often built through repeated interactions, shared experiences, and established reputations (Chopra and Wallace, 2003a).

Trust in Technology: refers to the confidence that users or organizations place in technological systems to perform reliably, securely, and in accordance with expectations. It emerges when confidence is placed in software, platforms, or technical infrastructures to perform reliably, securely, and according to specification. Examples include blockchain platforms, access control systems, and IoT networks. It involves beliefs about *reliability, functionality, and helpfulness* (Mcknight and Clay, 2011).

Across various domains, empirical studies confirm that trust in technology is a prerequisite for successful collaboration in socio-technical environments. When systems are perceived as reliable, functional, and helpful, they foster a sense of control and predictability that encourages organizational actors to delegate tasks, share information, and coordinate decisions through digital infrastructures. Conversely, when such trust is absent, even highly advanced technologies may fail to achieve their intended collaborative or efficiency benefits.

Digital Trust: emerges in inter-organizational processes where technology plays a role of mediator (Ritter, 2020). It is closely related to algorithmic and technology-mediated trust (e.g., blockchain). It captures trust in digital systems that facilitates interaction and transactions between stakeholders, emphasizing transparency, accountability, data integrity and ethical decision-making.

In IOBPS, digital trust involves a belief that interactions and transactions are secure, that all parties are identified, and that commitments can be traced and enforced. Digital trust builds upon trust in technology, as users must first rely on the underlying technological systems before engaging in collaborative processes. Specifically, it encompasses three key factors: security, ensuring data and transactions are protected from malicious interference; identifiability, confirming that participants are who they claim to be; and traceability, allowing contractual obligations and actions to be effectively monitored and enforced (Yan and Holtmanns, 2008). In practice, achieving digital trust is challenging due to limited transparency regarding platform assurance, making these factors critical to the design and evaluation of blockchain-based solutions in IOBPS.

Conceptual Ambiguities and Fragmentation

The literature also reveals conceptual ambiguities related to trust. Definitions are frequently domain-specific, and similar phenomena are described using different terminologies. Several studies implicitly conflate trust with related notions such as trustworthiness, confidence, or security, without clearly delineating conceptual boundaries.

Moreover, trust is often introduced as a contextual assumption rather than an explicitly defined construct. In such cases, trust is inferred from the presence or absence of mechanisms addressing uncertainty, risk, or opportunistic behavior. These observations indicate a fragmented conceptual landscape in which trust is conceptualized indirectly through associated concerns and challenges.

4.2.3 Results for RQ_2: Trust-Related Issues and Technological Approaches

This subsection reports the recurring trust-related issues identified of SCMS, and the technological mechanisms proposed to address them.

Recurring Trust Issues identified in SCMS

To identify trust issues, we focused on SCMS as a representative domain of IOBPs. SCMS aims to ensure that goods and services are delivered to consumers promptly, cost effectively, and efficiently (Giannakis et al., 2004). In SCMS, trust plays an important role. Tradelens (Jensen et al., 2019) is one example of blockchain in SCMS. It provides transparency, efficiency, and accountability in global trade by digitizing and streamlining the flow of information and documents among supply chain participants. Another example of practical applications is traceability of drugs that can be provided through a blockchain solution(Kambilo et al., 2022).

A Trust issue refers to a challenge, problem, or disagreement that affects the level of trust between individuals or parties (Kamble et al., 2020). They can be grounded on explicit evidence (frauds, contract violations, bad user experience) or on implicit beliefs. They are subjective and hard to grasp.

The reviewed studies identify a wide range of trust-related issues that undermine IOBPs. These issues frequently arise from limited transparency, lack of accountability, data integrity concerns, privacy vulnerabilities, and difficulties in tracing actions or transactions across multiple stakeholders.

Additional issues relate to organizational and human factors, including reluctance to share information, fragile stakeholder confidence, communication gaps, and resistance to technological adoption. Economic and technical constraints, such as scalability limitations, interoperability challenges, and high implementation or transaction costs, are also reported as barriers affecting trust.

Taxonomy of Trust Issues in SCM

Based on the scoping review, and in order to provide a systematic overview of the recurring trust-related issues undermining trust in supply chain systems, we consolidated the findings of twenty peer-reviewed studies addressing blockchain and digital technologies into seven categories. **Security, Data Governance, Data-related, User Acceptance, Traceability, Cost, and Auditability**

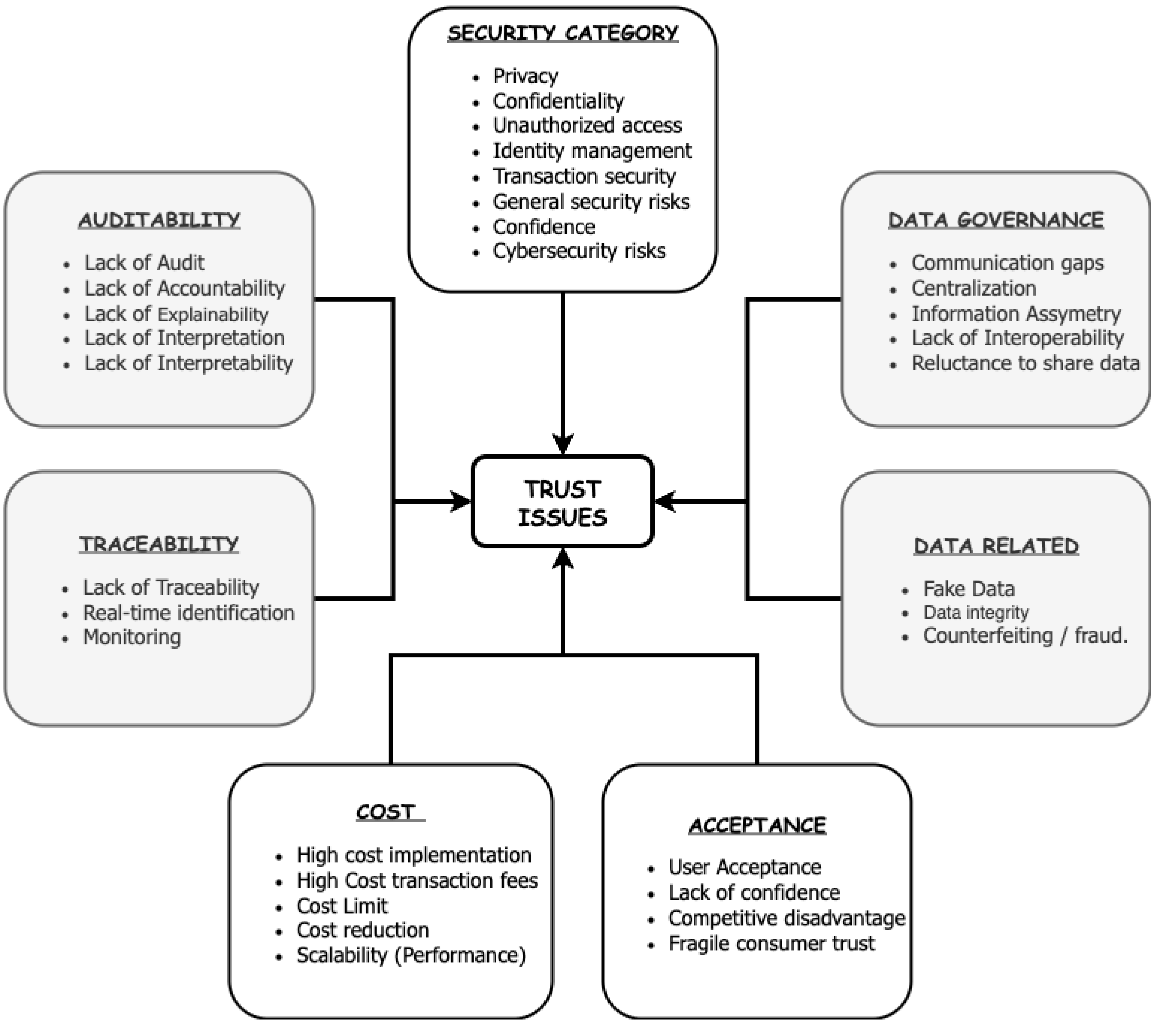

FIGURE 4.1: Taxonomy of trust issues in IOBPs

Security groups cover all trust issues related to the protection of digital assets, and system integrity. Item such as, unauthorized access, general security risks, transaction security, identity management, and privacy issues all point to vulnerabilities in how information and actors are authenticated, protected, and controlled. These are essential to trust because breaches in confidentiality, integrity, or authentication undermine system reliability.

Data-related issues concern the intrinsic properties of information itself. This includes, lack of transparency, data integrity issues, and unverified or fake data all highlight risks in the accuracy, availability, and quality of Information. Trust in supply chains depends on if data is timely, reliable, and resistant to manipulation, making these issues fundamentally data-centric.

Cost issues are economic barriers that influence the feasibility of adoption. High implementation costs, high transaction fees, scalability challenges, and the pressure for cost reduction all represent financial or resource constraints that organizations face. These costs not only limit technological deployment but also affect the long-term sustainability of blockchain-based solutions.

User acceptance issues reflects the human and organizational dimension of trust. Fragile consumer trust, lack of confidence, and competitive disadvantages linked to information sharing highlight how stakeholders perceive and accept technological changes. Even if the systems are technically robust, adoption will fail without stakeholder willingness and confidence in their fairness and benefits.

Traceability issues are tied to the ability to follow products throughout their life-cycle. Lack of traceability, real-time identification, and the persistence of counterfeiting or fraud demonstrate that without reliable tracking, products can be substituted or falsified. Strengthening traceability is therefore important to ensure authenticity and maintaining trust across supply chain transactions.

Data Governance issues relates to how information is managed, shared, and structured between actors. Information asymmetry, communication gaps, reluctance to share data, interoperability problems, and centralization risks all indicate weaknesses in collective data management. Without clear rules and distributed access, trust collapses due to unequal visibility, siloed systems, or dependence on a single authority.

Auditability focuses on the ability to monitor, verify, and explain system operations. Insufficient accountability and the lack of explainability or interpretability limit the capacity to assign responsibility and to provide transparency to oversight bodies. Effective auditability mechanisms are crucial for ensuring compliance, building trust, and enabling corrective actions when failures occur.

TABLE 4.2: Consolidated trust issues identified across 20 papers (P1–P20). A ✓indicates explicit mention.

Trust Issue (normalized)	**P1**	**P2**	**P3**	**P4**	**P5**	**P6**	**P7**	**P8**	**P9**	**P10**	**P11**	**P12**	**P13**	**P14**	**P15**	**P16**	**P17**	**P18**	**P19**	**P20**
Lack of **Transparency**	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓		✓				✓			
Insufficient **accountability / auditability**				✓						✓	✓	✓	✓	✓						
Data integrity	✓	✓	✓	✓	✓	✓	✓	✓		✓									✓	✓
Privacy Issues	✓	✓	✓	✓	✓	✓	✓		✓			✓								✓
Scalability (Performance)	✓	✓	✓			✓	✓										✓	✓		✓
Lack of **Interoperability**	✓		✓		✓	✓	✓		✓	✓		✓								
High costs (complexity)	✓		✓			✓	✓	✓	✓	✓					✓		✓	✓		✓
Lack of **Traceability**	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓								✓	✓
Unauthorized access		✓	✓		✓	✓	✓					✓			✓					
Information asymmetry	✓	✓	✓	✓		✓	✓	✓	✓	✓		✓								
Centralization				✓		✓		✓				✓	✓		✓					
Unverified **data accuracy** (fake data)	✓	✓		✓		✓														✓
Counterfeiting/fraud	✓			✓		✓	✓		✓											
Communication gaps between stakeholders					✓								✓		✓					
Security (generals) risks					✓		✓		✓						✓			✓		✓

(continued on next page)

Trust Issue (normalized)	**P1**	**P2**	**P3**	**P4**	**P5**	**P6**	**P7**	**P8**	**P9**	**P10**	**P11**	**P12**	**P13**	**P14**	**P15**	**P16**	**P17**	**P18**	**P19**	**P20**
Transaction security															✓					
High transaction costs (fees)																				✓
High costs / need for cost reduction	✓								✓				✓		✓		✓	✓		
Real-time identification / monitoring																			✓	
Identity management / authentication															✓					
Explainability / interpretability											✓									
Fragile **consumer trust** / lack of confidence	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓			✓	✓						
Reluctance to share data / information silos										✓		✓	✓							

Table 4.2 summarizes the conceptual resolutions of trust issues in SCMS with and the corresponding terms used in the reviewed papers (P1–P20).

These trust issues not only reveal the complexity of collaborative environments but also set the stage for RQ_2 by highlighting where technological mechanisms—such as blockchain—may offer structured, transparent, and reliable means to mitigate these vulnerabilities.

Technological Approaches to Address Trust Issues

Trust issues in IOBPs can be addressed through a combination of organizational, technical, and procedural strategies. The literature reviewed highlights a wide range of approaches at mitigating trust-related issues in collaborative settings, particularly within supply chain management. These approaches differ in their underlying assumptions about how trust is established, maintained, and enforced across multiple stakeholders.

Overall, the literature reveals **two distinct conceptual paradigms** for addressing trust issues in IOBPs. The first paradigm relies on **traditional (non-blockchain) approaches**, which emphasize relational, procedural, and centralized mechanisms. Within this paradigm, trust is fostered through relational governance mechanisms such as communication, shared values, and cultural alignment among partners (Pournader et al., 2020). Trust is also supported by centralized systems including shared databases, enterprise resource planning (ERP) platforms, and monitoring tools, which aim to improve information visibility and coordination (Lambert and Cooper, 2000). In addition, third-party assurance mechanisms such as audits, certifications (ISO standards), and legally binding contracts are employed to enforce compliance and reduce opportunistic behavior (Durugbo and Al-Balushi, 2023).

Despite their widespread use, these traditional approaches share important limitations. They depend heavily on partners' willingness to disclose the accurate information and on periodic validation by trusted intermediaries. They remain vulnerable to human error, fraud, information asymmetry, and intentional data manipulation. The reliance on centralized authorities and subjective assessments limits their ability to provide real-time, verifiable records (data integrity), and process compliance across distributed networks of stakeholders.

The second paradigm identify in the literature is formed by **blockchain-based approaches**, which propose a different conceptualization of trust. Rather than relying on trust in individual partners or intermediaries, blockchain shifts trust toward the system itself by leveraging decentralization, cryptographic verification, and consensus mechanisms. In this paradigm, periodic audits are replaced by continuous, real-time transparency enabled through immutable and shared ledgers (Kamble et al., 2020). Furthermore, smart contracts allow for the automated enforcement of rules and agreements, reducing subjective interpretations, human errors and intermediaries.

By addressing the technical vulnerabilities and coordination limitations inherent in traditional trust mechanisms, blockchain-based approaches aim to complement and in some cases substitute for organizational and relational forms of trust. This shift highlights blockchain's potential contribution to improving accountability, data integrity, traceability, and transparency, and trustworthiness in IOBPs.

4.3 Discussion: Answering RQ_1 and RQ_2

This section synthesizes and interprets the results presented in the previous section to answer the two main research questions (RQs). Rather than reiterating descriptive findings, the discussion integrates insights across domains to clarify how trust is conceptualized in the literature and how blockchain-based mechanisms are positioned in relation to trust-related issues in inter-organizational business processes.

4.3.1 Answering RQ_1: How Is Trust Conceptualized in the IOBPs?

The scoping review indicates that trust is not conceptualized as a single, stable construct in the literature on IOBPs. Instead, trust is framed through multiple, overlapping perspectives that reflect the socio-technical complexity of collaborative environments.

Across domains, trust is associated with the management of uncertainty and vulnerability; however, its object varies. In social and organizational perspectives, trust is primarily conceptualized as a relational expectation regarding the behavior, integrity, or competence of partners. From institutional perspective, trust shifts toward formal structures, rules, and governance mechanisms that constrain opportunistic behavior. In technology-oriented perspectives, trust refers to confidence in systems, infrastructures, and automated processes.

These perspectives are rarely unified within a single conceptual model. Instead, the literature tends to conceptualize trust implicitly through the dimensions it emphasizes, such as reliability, transparency, accountability, or predictability, rather than through explicit theoretical integration. As a result, trust appears as a layered construct combining social, technological, and digital elements, whose relative importance varies depending on context.

A key observation is that trust is often inferred from the presence or absence of mechanisms designed to mitigate risk and uncertainty. In this sense, trust is not only defined directly, but also indirectly conceptualized through the issues it seeks to address. This explains the prevalence of fragmented definitions and the frequent conflation of trust with related notions such as trustworthiness, security, or confidence. Overall, the literature portrays trust as a multi-dimensional and context-dependent construct, lacking a unified conceptual foundation in IOBPs.

4.3.2 Answering RQ$_2$: How can blockchain technology contribute to addressing trust issues?

In response to trust-related issues identified in IOBPs, the literature increasingly positions blockchain-based technologies as trust-enabling mechanisms. These technologies are not presented as substitutes for trust, but rather as infrastructures that reconfigure how trust is established, distributed, and maintained among collaborating actors.

Blockchain technology are primarily associated with addressing issues related to transparency, data integrity, accountability, and traceability. By providing immutable and shared ledgers, blockchain systems are described as reducing information asymmetry and limiting the need for bilateral trust between partners. In this sense, trust is partially transferred from organizational actors to technical and institutional mechanisms embedded in the system.

However, the review also indicates that blockchain-based approaches address only a subset of trust-related issues. While they can strengthen trust in technology and support digital trust, they do not eliminate the need for social and organizational trust. Issues related to user acceptance, governance, cost, scalability, and interoperability remain largely unresolved or are shifted to other layers of the system.

Moreover, the literature often assumes that technical properties such as immutability or decentralization automatically generate trust, without sufficiently examining the conditions under which these properties are perceived as trustworthy by stakeholders. This argument highlights a critical conflation: it assumes that blockchain's technical guarantees automatically produce trust, thereby overlooking the essential socio-organizational dimensions of partnership.

From Trust Issues to Design Requirements

The scoping review reveals that trust issues identified in inter-organizational business processes are inherently subjective and closely tied to stakeholder intentions, expectations, and perceptions. Addressing such issues through blockchain-based solutions therefore requires their translation into explicit and actionable system requirements.

Requirements Engineering provides a well-established foundation for this translation by enabling the systematic elicitation and formalization of stakeholder needs and concerns. In particular, ISO/IEC standards emphasize the importance of grounding system requirements in stakeholder goals and organizational contexts.

These observations motivate the need for a structured process that transforms trust-related issues, as identified in the literature, into trust requirements that can guide blockchain design decisions. The definition of such a process, together with its application and validation through a running example, constitutes the focus of the next chapter.

Limits of Blockchain Features and the Role of Blockchain Patterns

Although blockchain platforms provide core features such as immutability, distributed consensus, and cryptographic mechanisms, the literature shows that these features alone do not address all the trust issues identified in RQ_2. In inter-organizational business processes, trust issues often relate not only to data integrity, but also to how actors behave, share information, and coordinate their actions. Even when transactions are recorded on a blockchain, the reliability of the system still depends on how data are produced, who is responsible for submitting them, and how exceptions are handled. As a result, blockchain features mainly support technical aspects of trust, such as traceability and auditability, while other trust concerns remain outside their direct scope.

In practice, blockchain features are therefore combined with blockchain patterns that define how these features are used within concrete organizational settings. Blockchain patterns capture recurring ways of organizing interactions between actors, managing data flows, and defining responsibilities across organizational boundaries. Rather than focusing on technology alone, patterns make explicit the assumptions, rules, and coordination mechanisms required for collaboration. In the running example, trust is not achieved simply by storing events on a distributed ledger, but by applying appropriate patterns that determine who can write data, who can verify them, and how information is shared across the supply chain. This confirms that trust emerges from the joint use of blockchain features, design patterns, and organizational arrangements.

4.3.3 Cross-Analysis and Implications

Taken together, the answers to RQ_1 and RQ_2 highlight a misalignment how trust is conceptualized and how it is operationalized in existing research. While trust is acknowledged as a socio-technical construct, technological solutions, particularly blockchain-based ones tend to focus narrowly on technical dimensions of trust.

This gap suggests that effective trust support in inter-organizational business processes requires an integrated perspective that accounts for social, organizational, and technological dimensions simultaneously. The absence of such integration explains why trust-related issues persist despite the adoption of advanced digital infrastructures.

From a research perspective, these findings justify the need for a structured framework that explicitly links trust conceptualizations to technological mechanisms and design decisions. Rather than treating trust as an implicit outcome of technology adoption, trust should be addressed as a design concern grounded in clearly articulated conceptual foundations.

The discussion establishes that trust in inter-organizational business processes is multidimensional and only partially addressed by existing blockchain-based approaches. To illustrate how these insights can be operationalized and to assess their explanatory coherence, the next section introduces a running example that instantiates the identified trust dimensions and issues within a concrete inter-organizational scenario.

4.4 Running Example and Validation

This section introduces a running example to illustrate how the conceptual findings derived from the scoping review can be instantiated in a concrete inter-organizational business process. The objective is not to evaluate performance or to validate a technological implementation, but to assess the conceptual consistency, plausibility, and explanatory power of the results presented in Sections 4.2 and 4.3.

4.4.1 Running Example: Supply Chain Collaboration

We rely on the pharmaceutical on supply chain traceability running example introduced in Chapter 3. This example focuses on the delivery of a high-value, temperature-sensitive cancer medication from its manufacturer to a hospital pharmacy, involving multiple autonomous actors, including logistics providers, intermediaries, and regulatory authorities

4.4.2 Instantiating Trust Conceptualizations

Based on the synthesis of RQ_1, trust in the running example can be instantiated along three complementary dimensions.

Social Trust: manifests through long-term partnerships, reputational considerations, and expectations regarding partners' integrity and competence. For instance, manufacturers rely on logistics providers to handle goods responsibly, while distributors expect accurate and timely information from upstream actors.

Trust in technology arises from reliance on shared digital systems used to record and exchange operational data. Actors must trust that these systems function reliably, preserve data integrity, and enforce access control policies consistently across organizational boundaries.

Digital Trust: emerges at the intersection of organizational collaboration and digital mediation. It reflects confidence that transactions and process executions are transparent, traceable, and auditable, and that digital records can be relied upon for compliance verification or dispute resolution.

This instantiation illustrates how trust in inter-organizational business processes is inherently multi-dimensional and cannot be reduced to a single referent.

4.4.3 Mapping Trust Issues to Blockchain Mechanisms

The trust-related issues identified in RQ_2 can be directly mapped onto the running example. Common challenges include limited transparency across supply chain stages, information asymmetry between actors, difficulties in tracing product provenance, and concerns regarding data manipulation or accountability.

Blockchain-based mechanisms discussed in the literature are frequently proposed to address these issues by providing shared, immutable records of transactions and events. In the running example, such mechanisms can support traceability by recording product movements, enhance transparency by granting controlled visibility to authorized actors, and improve auditability by enabling verifiable process histories.

However, consistent with the discussion in Section 4.3, these mechanisms primarily address technological and digital dimensions of trust. Social trust remains dependent on organizational behavior, governance arrangements, and stakeholder acceptance. The running example thus confirms that blockchain technologies can support trust, but do not replace the need for social and institutional coordination.

4.4.4 Validation Through Conceptual Consistency

The running example serves as a form of conceptual validation by demonstrating that the trust dimensions, issues, and mechanisms identified in the scoping review can be coherently instantiated within a realistic inter-organizational scenario. The example shows that the proposed categorization of trust forms and issues aligns with observed collaborative practices and that the role attributed to blockchain-based mechanisms is consistent with their representation in the literature.

Rather than validating effectiveness, this section validates conceptual alignment between theory, observed challenges, and proposed technological responses. This alignment supports the relevance of the findings and provides a credible foundation for the design-oriented contributions developed in subsequent chapters.

4.5 Conclusion

This chapter set out to clarify how trust is addressed in the literature on inter-organizational business processes, with a particular focus on the Supply Chain Management domain. Through a scoping review, it systematically analyzed how trust is defined, operationalized, and supported by existing approaches, including but not limited to blockchain-based solutions. Rather that assuming a unified understanding of trust, the reviewed revealed a landscape characterized by heterogeneous conceptualizations, implicit assumptions, and fragmented analytical perspectives.

A first outcome of this chapter is the observation that trust is rarely treated as an explicit, first-class construct. Instead, it is often approached indirectly through related issues such as transparency, traceability, data integrity. While these dimensions capture important aspects of trust, their isolated treatment obscures the broader IOBPs. As a result, trust tends to emerge as an implicit outcome of technical or organizational mechanisms rather than as a design objective in its own right.

The review also highlighted a recurring set of trust-related issues that persist across domains and technological choices. These include limited visibility across organizational boundaries, information asymmetry between actors, difficulties in establishing reliable product provenance, and concerns regarding opportunistic behavior or data manipulation. Although blockchain technologies are frequently proposed as a response to these issues, the literature often adopts a solution-driven stance, emphasizing technical capabilities without sufficiently analyzing how and under which conditions these capabilities contribute to trust at the process level.

Overall, this chapter motivates the central thesis of this work: addressing trust in inter-organizational business processes requires structured design frameworks that make trust explicit, model its dependencies, and support trade-off analysis among competing soft goals. These findings provide the conceptual foundation for the subsequent chapters, which introduce a design-oriented approach that integrates trust modeling, knowledge representation, and multi-criteria decision support to guide the selection and configuration of blockchain-based patterns.

Chapter 5

Addressing Trust in Supply Chain Management through Blockchain Software Patterns

The previous chapter established that trust in IOBPs is a multifaceted concept, expressed through heterogeneous trust issues across domains and application contexts. While blockchain technologies are frequently proposed in the literature of as trust-enabling solutions, Chapter 4 indicates that trust-related issues are rarely translated into explicit, actionable design decision.

This Chapter addresses this gap by proposing a systematic, requirements-driven approach for addressing trust issues in blockchain-based systems. Building on the trust issues identified through the scoping review, the chapter introduces a structured process that transforms trust issues into trust requirements and maps these requirements to concrete blockchain design solutions in the form of software patterns.

The objectives of this chapter are threefold. First, it provides a method for translating subjective and implicit trust issues into explicit system-level requirements. Second, it establishes a mapping between requirements and blockchain mechanisms, highlighting both the capabilities and limitations of intrinsic blockchain features. Third, it demonstrates the applicability of the proposed approach through a running example drawn from a pharmaceutical supply chain traceability scenario, ensuring continuity with the running example introduced in Chapter 3.

This chapter constitutes a core contribution of the thesis by moving from conceptual analysis toward design-oriented knowledge that supports the construction of trustworthy inter-organizational systems.

5.1 Introduction

5.2 From Trust Issues to Trust Requirements in SCM

Trust issues are subjective and sometimes stem from stakeholders' intentions. Using Blockchain-based technologies implies to define clear requirements that address these issues. Defining requirements starts with understanding the stakeholders' intentions, needs, goals, or objectives, as outlined in ISO (documentation, 2018).

Requirements Engineering (RE) serves as a bridge between stakeholders and system developers (Pohl, 2010). In the context of trust, RE enables the transformation of subjective trust issues into explicit, actionable system requirements by analyzing stakeholder expectations, examining existing processes, and formalizing trust-related concerns at the system level.

In the context of this thesis, operationalizing trust means making trust-related expectations explicit in the form of (trust) requirements that can guide architectural and design decisions. These requirements act as an intermediary layer between abstract trust issues and concrete technical mechanisms, such as blockchain features or software patterns.

5.2.1 Taxonomy of Trust Requirements in SMCS

By applying the process described above to the trust issues identified in Chapter 4 results in a structured set of trust requirements. To provide analytical clarity and design support, these requirements are organized into a taxonomy reflecting recurring trust issues in SCMS.

The taxonomy groups trust requirements into seven categories: Security, Data Governance, Data-related, Traceability, Cost Control, Auditability, and User Acceptance. Each category captures a distinct dimension of trust expressed as **soft goal**, while remaining interrelated with others.

Security-related requirements address confidentiality, integrity, privacy, and protection against unauthorized access. Data governance requirements concern how data is shared, controlled, and interoperable across organizational boundaries. Data-related requirements focus on data quality, authenticity, and reliability. Traceability requirements ensure the ability to track products, transactions, and events across the process lifecycle. Cost control requirements reflect economic constraints associated with system operation. Auditability requirements support monitoring, accountability, and compliance. Finally, user acceptance requirements capture stakeholders' confidence and willingness to rely on the system.

This taxonomy serves as a design abstraction that bridges trust issues and technical solutions. Rather than prescribing specific technologies, it structures trust concerns in a way that supports systematic design reasoning.

5.2.2 Translating Trust Issues Into Trust Requirements

Following ISO/IEC guidelines (documentation, 2018) and established RE practices (Pohl, 2010), transforming trust issues into trust requirements involves systematically analyzing stakeholder expectations and current organizational practices. Subjective trust concerns are progressively refined into objective requirements that can guide system design.

To this end, the present study adopts a Requirements Engineering–based approach and proposes a systematic process for translating identified trust issues into trust-related requirements, ensuring traceability between stakeholder concerns and technical design decisions. We propose a **systematic process** to transform trust issues into requirements through the following steps:

(1) Elicitation of user trust issues: In this step, the evidences of trust issues has to be gathered. Various techniques defined in the fields of requirements engineering and knowledge management can be used to collect the empirical data, including interviews, case studies, workshops, action research, etc. The outcome is a collection of trust issues expressed by end users or stakeholders. In this work, the data about trust issues has been collected through the literature review.

Trust issues are elicited through the scoping review of the literature. Different studies often describe similar issues using heterogeneous formulations. For example, accountability-related concerns are expressed in (Hasan et al., 2020) through the need to clarify responsibilities and interactions between supply chain actors to ensure transparency, while (Jayashri et al., 2023) frames the same issue in terms of accountability and incentive mechanisms designed to promote trustworthy behavior. Despite differences in wording, both sources highlight accountability as a central trust issue.

(2) Analysis: The purpose of this step is to analyze each expressed trust issue in order to identify a subject of trust (actor, system, process, technological component), an object of trust (e.g., data, activity, function, etc.) and an expected relationship that must be established between the former and the latter in order to mitigate the issue. For example: Lack of accountability issue addresses a business partner in the supply chain (the subject) and the transaction data (the object). The issue expresses a trustor's belief that, in case of dispute, the partner can avoid responsibility for his actions unless the formal proof of such actions is provided. To mitigate the issue, the transaction has to be non-reputable (the relationship). In the field, such analysis has to be conducted iteratively, confirming and validating the results with users.

(3) Specification of requirements: In this step, the requirements are documented based on the analysis from the previous step and following the recommendations from (documentation, 2018). The outcome of this step is a formalized requirement specification. Example: For the Lack of accountability (I1) issue, we formalize the corresponding trust requirement as follows: "System must guarantee non-repudiation of data".

Table 5.1 presents the mapping between the trust issues identified in the literature, the corresponding references, and the formalized trust requirements.

TABLE 5.1: Taxonomy & Mapping trust issues into trust Requirements

Taxonomy of Trust Issues		**Trust issues to Trust requirements**	
Issue	**References**	*Requirement Specification*	*Requirement Name*
Lack of accountability (I1)	(Hameed et al., 2022; Jayashri et al., 2023; Manning et al., 2022Wei, 2020; Casino et al., 2021; Caro et al., 2018; Kuhn et al., 2021; Kamble et al., 2020; Chang and Chen, 2020)	System must guarantee non repudiation of data.	accountability
Lack of auditing(I2)	(Imeri et al., 2019; Hameed et al., 2022; Kuhn et al., 2021; Chang and Chen, 2020)	System must have log files to facilitate auditing	Audit
Lack of security (I3)	(Imeri et al., 2019; Xu and Choi, 2021; Biswas et al., 2023; Wei, 2020; Casino et al., 2021; Hasan et al., 2020)	System must guarantee a no intrusion of external stakeholders	Security
Lack of interoperability(I4)	(Imeri et al., 2019; Xu and Choi, 2021; Masudin et al., 2022; Wei, 2020; Casino et al., 2021; Figorilli et al., 2018; Casino et al., 2019; Baralla et al., 2021)	System must be flexible to insert other modules or SI.	Interoperability
Lack of transparency(I5)	(Hameed et al., 2022; De Giovanni, 2020; Manning et al., 2022; Kuhn et al., 2021; Hasan et al., 2020; Kamble et al., 2020; Baralla et al., 2021)	System must save and share data in readable and transparent manner for everyone.	Transparency
High cost of transaction(I6)	(De Giovanni, 2020; Xu and Choi, 2021; Biswas et al., 2023; Wei, 2020; Figorilli et al., 2018; Baralla et al., 2021)	System doesn't cost much by utilization	Cost control
Lack of reliability(I7)	(Hameed et al., 2022; Jayashri et al., 2023)	System must be reliable	Reliability
Lack of confidence(I8)	(Jayashri et al., 2023)	System must guarantee data confidence	Confidence
Lack of confidentiality(I9)	(De Giovanni, 2020; Kamble et al., 2020)	System must guarantee process confidentiality	Confidentiality
Lack of privacy(I10)	(De Giovanni, 2020; Biswas et al., 2023; Hasan et al., 2020; Casino et al., 2019; Kamble et al., 2020; Chang and Chen, 2020)	System must guarantee data privacy	Privacy
Lack of responsibility(I11)	(Manning et al., 2022)	System must save a trace of each transaction	Responsibility
Lack of traceability(I12)	(Masudin et al., 2022; Biswas et al., 2023; Baralla et al., 2021)	System must guarantee tracking of data	Traceability
Lack of direct communication(I13)	(Hameed et al., 2022; Jayashri et al., 2023)	System must be decentralised with Peer-to-Peer communication	Decentralized
Lack of RT identification(I14)	(Masudin et al., 2022; Casino et al., 2021; Caro et al., 2018; Hasan et al., 2020; Chang and Chen, 2020)	System must track data on real time	R-T identification
Lack of data integrity(I15)	(Masudin et al., 2022; Wei, 2020; Casino et al., 2021; Caro et al., 2018; Kamble et al., 2020; Chang and Chen, 2020)	System must be guaranteed data integrity	Integrity
Lack of User acceptance(I16)	(Biswas et al., 2023; Caro et al., 2018)	System must be guaranteed a User satisfaction.	User Acceptance
Lack of monitoring(I17)	(Figorilli et al., 2018; Kuhn et al., 2021; Casino et al., 2019; Baralla et al., 2021)	System must give the possibility to monitor all processes	Monitoring
Fake data(I18)	(Biswas et al., 2023; Casino et al., 2019)	System must guarantee data origin	Fake Data
Lack of scalability(I19)	(Wei, 2020; Kuhn et al., 2021; Kamble et al., 2020; Chang and Chen, 2020)	System must guarantee a High availability of data	Scalability
Lack of tamper-proof(I20)	(Wei, 2020; Figorilli et al., 2018; Kuhn et al., 2021; Kamble et al., 2020)	System must guarantee data tamper-proof	Tamper-proof
Lack of good storage(I21)	(Caro et al., 2018; Kuhn et al., 2021; Hasan et al., 2020; Casino et al., 2019)	System needs Good support for storage of data	Storage

5.3 Use of Blockchain Software Patterns for Meeting Trust Requirements in SCM.

Blockchain is often considered a de facto trust enabler. We argue that, while offering a number of key features, stock blockchain may not provide a complete solution for the specific trust requirements in a given context. In this section, we discuss the current limitations of stock blockchain solutions and propose the use of blockchain software patterns to efficiently address the specific trust requirements in SCM.

According to the literature, intrinsic features of blockchain(public) technology address a number of requirements in SCM including trustworthiness. However, this technology also has limitations that can overshadow the benefits and have a negative impact on trust. These limitations have to be taken into account when making *design decisions*. Challenges related to privacy, scalability, trust and interoperability are some examples relevant to the SCM domain. These challenges can be efficiently addressed using specific blockchain software patterns. Here are some examples.

Blockchain is not suitable for storing and managing large amounts of data. Keeping images and other large data sets in blockchain can be expensive due to the high cost of transaction fees. This undermines the performance and credibility of the solution and negatively impacts user's trust in this technology. To overcome this, off-chain data storage (patterns) such as the Interplanetary File System (IPFS) can be implemented in conjunction with blockchain. Using blockchain to store hashes of the data and off-chain storage to store extensive data can reduce transaction costs and add scalability to the system.

Along the same lines, if a blockchain solution is not designed with security in mind, it can be vulnerable to cyberattacks or other forms of malicious activity. This can result in the loss of funds or the compromise of sensitive information. Additionally, if the consensus mechanism used in the blockchain is not well adapted, it could lead to issues with trust in the network. For privacy, if a blockchain is not designed with privacy in mind, it can lead to the exposure of sensitive information because ledger is public for all stakeholders. The use of encryption-on chain(patterns) data is recommended to address these challenges.

Table 5.2 presents the mapping between the trust requirements in SCM, the key blockchain features discussed in Section 2.2.1 that are recurrently used to meet these requirements, and the blockchain software patterns from Six, Herbaut, and Salinesi, 2022b, which we identify to complement the features. We consider three cases:

CASE 1 blockchain features provide a complete solution for specific requirements. For example: transparency, as the transparent ledger offered by blockchain inherently meets this requirement without the need for additional patterns (indicated with a ✓ symbol in BC features column in the table).

CASE 2 : blockchain features are insufficient to meet a requirement and blockchain software patterns are proposed. For example, a public blockchain's (transparency) cannot ensure data privacy. In this case, a private blockchain or encryption on-chain data patterns can encrypt data during transit to maintain privacy on the blockchain (indicated with a – symbol in BC software patterns column in the table).

TABLE 5.2: Mapping of Trust requirements on Blockchain features and patterns
✓ satisfied, ! partially satisfied, – unsatisfied

Requirement	**Blockchain Features**	**Blockchain Software Patterns Six, Herbaut, and Salinesi, 2022b**
Accountability	✓ Accountability	✓ Identifier Registry
Audit	✓ Audit	–
Interoperability	–	✓ Contract Observer
Transparency	✓ Transparency	–
Cost control	! Automating	! Minimize on-chain data, Flyweight, Off-chain storage
Reliability	✓ Immutability	–
Privacy	–	✓ On-chain encryption
Responsibility	–	✓ Identifier Registry
Traceability	✓ Traceability	–
Scalability	–	✓ State channel (50%), off-chain storage (50%)

CASE 3 : blockchain features offer only a partial solution and can be complemented by using software patterns to meet a requirement. For example, the automation of information systems in blockchain can lead to increased costs if the data transit is extensive in terms of storage. To address this issue, minimize On-chain data or using Flyweight patterns can limit the data size in each transaction and reduce costs (indicated with a ! symbol in the table).

Implementing blockchain solutions by using software patterns can help to improve scalability, good storage, privacy and trust in the supply chain management environment. In Table 5.2, we present the mapping of 16 out of 21 trust requirements in SCM .

According to our analysis, Security, Confidence, Confidentiality, Fake data, and User acceptance requirements are not fully met by the current blockchain solutions (features and/or patterns). These complex problems require more research and development and need to be addressed by the blockchain community in the future.

5.4 Illustrative Example and Discussion

To illustrate the proposed mapping between trust-related issues and blockchain software patterns, and to provide an initial validation of the approach, this chapter relies on an illustrative example drawn from the food supply chain domain. The example focuses on the use of blockchain technology to support traceability for food safety risk management and regulatory compliance. It serves as a concrete instantiation of the concepts, mappings, and reasoning mechanisms introduced in this work.

5.4.1 Application of the Mapping Process to the Illustrative Example

By applying the process defined in sub-section 5.2.2, this chapter identifies a set of recurrent trust-related issues arising in the illustrative example and maps them onto the proposed taxonomy, as summarized in Table 5.1. These issues are discussed below.

- *Lack of Interoperability (I4):* The illustrative example highlights the need to support collaboration among heterogeneous stakeholders operating autonomous information systems. The objective is to enable interoperability across organizational boundaries while preserving stakeholder autonomy, which constitutes a key trust challenge in inter-organizational environments.
- *High Transaction Costs (I6):* Transaction costs emerge as a major concern due to the large volume of data generated and exchanged along the supply chain. To mitigate this issue, the example emphasizes the separation of on-chain and off-chain data management, where decentralized storage solutions are used to reduce on-chain transaction overhead.
- *Lack of Scalability (I19):* Scalability issues stem from the continuous growth of traceability data that must be securely stored and accessed. The example illustrates the necessity of decentralized storage mechanisms to ensure that the system can scale while preserving data integrity and availability.
- *Lack of Traceability (I12):* The example reveals that traceability information is typically fragmented, as each participant maintains its own traceability mechanisms and records. This fragmentation limits end-to-end visibility and undermines trust among stakeholders, particularly in the context of compliance verification and risk management.

- *Lack of Data Integrity and Privacy (I15, I10):* Finally, the illustrative example underlines the dual requirement of ensuring data integrity and preserving the privacy of transactions and involved actors. Smart contracts and cryptographic mechanisms are introduced as key enablers to guarantee trustworthy data sharing while protecting sensitive business information.

5.4.2 Results

In this example, we were able to identify the trust issues from the case text and to map them on our taxonomy. Though the patterns indicated by the case authors are not expressed explicitly, we were able to match them with the patterns provided by (Six, Herbaut, and Salinesi, 2022b). The authors in the case use IPFS to store vast amounts of data, which generates a hash secret on a smart contract address to reduce transaction costs and enable `Off-chain data storage`. The authors are also concerned with ensuring the privacy and traceability of data through encryption and the use of hash (`Encryption On-chain` & `Hash secret`).

The patterns used in the case correspond to our recommended blockchain software patterns for the following four requirements: Privacy, Integrity, Storage and Cost control. For the storage and Cost c requirement, we propose the `Limit storage` pattern to set a gas limit for transactions in addition to the `Off-chain data Storage`, already identified by the case authors.

The Traceability requirement does not require a specific pattern as it is directly provided by blockchain. This corresponds to our mapping in Table 5.2. The Interoperability requirement is not addressed in the paper. We propose the following blockchain software patterns to meet these requirements: the `Contract observer` pattern to guarantee interoperability and confirm that data written to the blockchain comes from a trustworthy source, and the `Identifier registry` pattern to track transactions and hold stakeholders accountable in the event of any problems.

In this example, we used a mapping proposed in the previous section to extend the solution proposed by the authors with two specific blockchain software patterns. This proposal completes the solution by addressing more trust requirements.

5.5 Discussion and Limitations

Software patterns provide developers with technical best practices. However the trust implications of the patterns are not explicit. Our literature analysis shows that trust

,

TABLE 5.3: Blockchain features and Patterns resolving trust issues
✓: Satisfied Req., !: partially satisfied Req.,−: Unsatisfied Req.

Requirements (extracted from the case)	BC intrinsic features	Patterns (extracted from the case)	BC software patterns (our proposal)
Interoperability	−	−	Contract Observer
Cost control	!	Off-chain data storage	Limit Storage
Privacy	−	Encryption On-chain	Encryption On-chain
Traceability	✓	−	−
Integrity	!	Hash Secret	Hash Secret
Storage	−	Off-chain data storage	Off-chain data storage, Limit storage

issues are not explicitly addressed in the design of the SCM solutions. Trustworthiness is often taken for granted by the mere use of a blockchain and cannot by validated.

The blockchain often meets trust requirements through a goal-oriented approach, which many researchers have explored. For example, authors in (Khalifa et al., 2019) focused on studying and identifying trust requirements in blockchain systems and created a trust engineering taxonomy to meet blockchain systems' trust requirements and goals. However, they did not provide evidence of how their taxonomy can be used or how to meet trust requirements using goals. In (Johng et al., 2020), a goal-oriented approach for business process reengineering is discussed. Here trustworthiness concerns are explicitly represented as (soft) goals and mapped to the relevant trust-enhancing features of blockchain, supporting business process reengineering. These works use a goal-oriented approach to focus on requirements and how specific blockchain features meet trust requirements.

The uniqueness of our approach resides in combining specific blockchain patterns from (Six, Herbaut, and Salinesi, 2022b) and intrinsic blockchain features to address trust issues and to create trustworthy solutions in the field of SCM. We make a first attempt to create a taxonomy of *trust issues* and *trust requirements* and define their design implications for blockchain solutions. This taxonomy will guide organizations to create trustworthy solutions in SCM.

Despite the promising results obtained from our study, it is important to note that there are several limitations and opportunities for improvement that should be addressed. These include:

- We identified trust issues using a sample of 18 research articles on SCM. We plan to conduct Systematic Literature Review to validate our findings and to extend our taxonomy.
- We limited our study to SCM, and more work is required to generalize this approach to other domains.
- Both trust issues and pattern were often not explicit in the literature, we had to rely on our expertise and interpretation to extract them. Our main effort is to promote standardization and the use of a common language (taxonomy of trust issues and requirements) to alleviate the interpretation bias in the future.

For the practical and effective use of the proposed taxonomy and mapping, we plan to establish a recommendation system application where developers can select the issues encountered as input and have access to the blockchain software patterns they can leverage to ensure a design that inspires trust for collaboration.

5.6 Threats to Validity

As with any design-oriented and illustrative research, this chapter is subject to several threats to validity. These threats are discussed according to internal, external, and construct validity.

5.6.1 Internal Validity

Internal validity concerns the soundness of the reasoning linking trust issues, blockchain features, and pattern selection. A first threat lies in the subjective interpretation involved in mapping trust requirements to blockchain mechanisms. Although these mappings are informed by prior literature and domain knowledge, they remain partially dependent on expert judgment.

Another potential threat is related to the illustrative nature of the running example. The observed alignment between trust issues and selected patterns may be influenced by the way the case is framed, rather than by intrinsic causal relations. To mitigate this risk, the chapter relies on explicit mapping tables and makes trade-offs and assumptions visible rather than implicit.

5.6.2 External Validity

External validity refers to the generalizability of the results beyond the considered use case. The chapter focuses on a food supply chain traceability scenario, which represents a common but specific class of inter-organizational systems. Consequently, the identified trust issues and recommended blockchain patterns may not fully apply to other domains with different regulatory, organizational, or technical constraints.

However, the objective of this chapter is not statistical generalization but analytical generalization. The contribution lies in demonstrating how the proposed approach can be instantiated in a realistic setting. Replicating the analysis in other supply chain contexts or inter-organizational domains would strengthen external validity and is left for future work.

5.6.3 Construct Validity

Construct validity concerns the extent to which trust and its related soft goals are adequately captured and operationalized. Trust is a multidimensional and abstract concept, and its decomposition into security, performance, and transparency-related soft goals may not cover all possible interpretations found in the literature.

Moreover, the impact of blockchain patterns on these soft goals is represented using qualitative or ordinal assessments rather than quantitative metrics. While this choice is consistent with early-stage design decision support, it may introduce ambiguity in the interpretation of results. To reduce this threat, the chapter maintains a clear separation between conceptual constructs and their operational representations and ensures consistency across the evaluation process.

5.7 Conclusion

While blockchain technology is often considered as the de facto trust enabler, some limitations persist. These limitations have to be systematically addressed by improved design practices. In this article we consider blockchain software design patterns to address the trust issues in SCM domain.

First, we provided an overview and developed a taxonomy of trust issues based on literature in the supply chain. Following the recommendations from requirement engineering, we translated the trust issues into explicit trust requirements. We examined the existing solutions that address these trust requirements in the literature and identified their limitations: We argue that the use of intrinsic features of blockchain often provides only a partial solution for trust issues in SCM. To complete this solution, we propose the use of blockchain software patterns.

We identified 12 patterns from the blockchain software pattern literature that can support the trust requirements in SCM and evaluated our proposal on one example from the literature. This preliminary evaluation shows the relevance of the trust issues taxonomy defined in this work. The proposed blockchain software patterns extend the solution from the case, demonstrating the potential interest and added value of our mapping.

This work aims at helping enterprises to better understand their trust-related requirements and to improve the design of their SCM systems.

Chapter 6

Technology-Aware Enterprise Modeling (TEAEM)

This chapter proposes a holistic design approach for aligning organizational objectives with blockchain-based technical decisions. The approach extends the traditional Model-Driven Architecture paradigm by incorporating bottom-up constraint analysis and constraint propagation mechanisms.

The chapter introduces a Unified Feature Model that captures both organizational goals and technical properties of the target solution. This model serves as a formal representation of design choices and their interdependencies. Model-checking techniques are then applied to verify consistency and to reason about the constraints emerging from the interaction between organizational and technical concerns.

Using a supply chain management running example, the chapter demonstrates how technical decisions specified at the feature level can be formalized, validated, and propagated as strategic constraints within a Goal Model. This process enables the systematic reconsideration and redesign of both business-level and technical enterprise models based on validated design limitations.

6.1 Challenging the Model-Driven Architecture Paradigm

6.1.1 Motivation and Positioning

The Model-Driven Architecture (MDA) provides a fundamental framework for the design and development of enterprise information systems, prioritizing the top-down design process. It introduces three levels of model abstraction and a model transformation process, ensuring a traceability and alignment between the high-level business goals and developed technological components (Mellor, 2004). In this methodology, the selection of technological platforms and components is deferred to lower levels of the MDA design hierarchy. This approach is justifiable under the assumption that technological choices have primarily local effects.

However, the emergence of disruptive technologies, such as blockchain challenges the established principles of MDA by exerting a broader influence on processes and goals of the enterprise as a whole (Marques and Pereira, 2023). MDA shows limitations when dealing with technology-specific constraints that directly impact organizational goals. Technical decisions are often introduced late in the design process, while their implications on business goals, governance, and trust-related issues remain insufficiently anticipated.

This chapter address this limitation by challenging the strictly top-down nature of traditional MDA and by introducing a design approach that explicitly accounts for technological constraints early in the enterprise modeling process.

6.1.2 From Model-Driven Architecture to TEAEM

Model Driven Architecture is a software design approach that provides a set of guidelines for structuring specifications expressed as models, supporting traceability and business-IT alignment (Mellor, 2004). MDA defines three primary types of models:

- *Computation Independent Model* (CIM) represents the system's requirements and business context without detailing the structure or processing;
- *Platform Independent Model* (PIM) specifies the system's structure and functionality but abstracts away the details of any specific implementation platform;
- *Platform Specific Model* (PSM) provides the technical details on system implementation using a particular technology or platform.

This layered organization has long provided a robust foundation for enterprise information system design. By linking conceptual business views to technical models through systematic transformations, MDA enables a structured reasoning across abstraction levels. Early contributions, such as (Wegmann and Preiss, 2003), proposed hierarchical design principles that connected systems engineering with enterprise architecture and influenced subsequent enterprise modeling approaches.

As research evolved, ensuring alignment between models across these layers became a central concern. Misalignment risks distorting business intent during its translation into technical implementations. To address this issue, numerous approaches have proposed method and tools to validate consistency between enterprise goals, business processes, and system designs (Cortes-Cornax et al., 2015; de la Vara et al., 2013; Gröner et al., 2014; R. Guizzardi and Reis, 2015; Horita et al., 2014; Ruiz et al., 2015; Wegmann et al., 2007). Formal verification technique further strengthened this line of work by enabling detection of inconsistencies (Rychkova, Irina, 2008).

More recently, researchers have sought to integrate organizational strategy and structure into the MDA lifecycle. (Pastor et al., 2021), introduced a semi-automated strategy-to-code approach, grounded in the LiteStrat Noel et al., 2021, emphasize traceability from strategic decisions to technical artifacts, supporting consistency and accountability across design levels.

However, modern enterprise systems increasingly rely on emerging technologies such as Artificial Intelligence, the Internet of Things, and blockchain. While these technologies offer new capabilities aligned with specific business objectives, they also introduce trade-offs related to scalability, cost, privacy, or governance (Marques and Pereira, 2023; Morkunas et al., 2019; Rajnak and Puschmann, 2021). Traditional MDA-based approaches remain largely technology-neutral and therefore offer limited support for reasoning about such trade-offs during early design stages.

The Technology-Aware Enterprise Modeling approach proposed in this thesis directly addresses this gap. TEAEM builds on the principles of MDA but extends them to explicitly incorporate the technical characteristics, constraints, and side effects of emerging technologies into the enterprise modeling process. By doing so, TEAEM ensures that strategic goals (trust or not) remain feasible in practice, even as they pass through the multiple layers of abstraction that separate business goals from their technical realization.

6.2 TEAEM: Technology-Aware Enterprise Modeling

TEAEM is a holistic design approach that extends the MDA paradigm. Where MDA follows a primarily top-down logic—flowing from business Goals to technical implementation, TEAEM adds a bottom-up constraints analysis and propagation. This means that technical realities and constraints identified at later design stages can feed back into earlier models, ensuring that strategic goals remain achievable in practice.

6.2.1 TEAEM Design and Modeling Principles

Figure 6.1 illustrates the TEAEM approach. We use goal modeling (Van Lamsweerde, 2001), feature modeling (Yu et al., 2008) and component modeling (Trinidad et al., 2007) to represent an enterprise solution at the three MDA

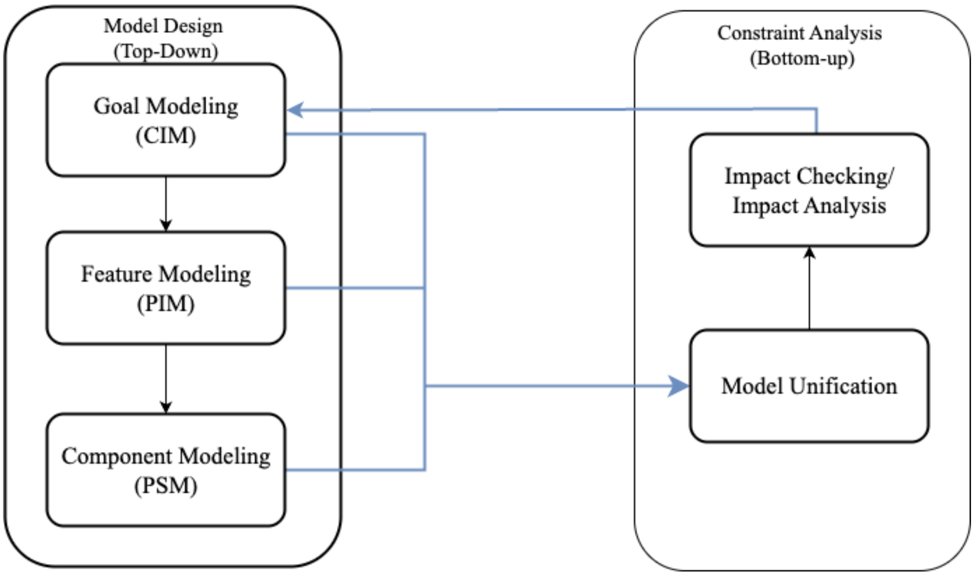

FIGURE 6.1: TEAEM Steps

Goal Modelling (CIM): A goal model representing the high-level business goals, requirements, and domain concepts is created in this step. This model corresponding to CIM - the highest level of abstraction within MDA. It outlines the dependencies between goals and intended outcomes ensuring alignment with broader organizational context. These dependencies are explored in the further constraint analysis.

Feature Modelling (PIM): A feature model is defined in this step. This model specifies the hierarchical structure of abstract technical functionalities (features) of the prospective solution and corresponds to PIM. The dependencies and constraints between the features define alternative configurations for the technical solution.

Component Modelling (PSM): A component model is created in this step. This model specifies technological components and platform-specific details necessary for implementation of the features defined in PIM. Technical constraints between the components provide the information about alternative implementations of the designed solution.

Unified Feature Model (UFM): In this step, business goals, technical features, and components are specified within a single Unified Feature Model. For further impact analysis, we use logical expressions to formalize the mappings between the model elements defined at different MDA abstraction levels. We define the following Mapping:

- *Goal to Feature Mapping* associates the business goals with the technical features satisfying these goals. This mapping reflects technical assumptions made by system engineers.
- *Feature to Goal Mapping* associates the technical features with the goals that can be possibly compromised or inhibited by implementation of these features.
- *Feature to Component Mapping* associates the features with the specific technological components implementing these features.
- *Component to Feature Mapping* explicitly defines possible restrictions or incompatibilities between the features and the components.
- *Component to Goal Mapping* indicates possible restrictions between the (soft) goals and the components such as quality, feasibility, performance issues etc. We combine these mappings with the constraints defined in the previous TEAEM steps to form the UFM.

Model Checking & Impact Analysis: We use model-checking to validate the Unified Feature Model (UFM) and to identify inconsistencies (if any) in its specification. A solver finds possible configurations of a solution and/or shows the conflicts between the elements of the unified model. We propose a domain-specific interpretation of these conflicts and recommendations for their resolution.
The Constraint Analysis can be performed at different stages of Model design and serve to:

- Identify and propagate the effect of design decisions specified in PIM on CIM;
- Identify and propagate the effect of design decisions specified in PSM on PIM and/or on CIM.

6.2.2 TEAEM in Practice: Implementation

To bring the TEAEM approach to practice, we developed a semi-automated tool that demonstrates the feasibility of the proposed steps. This includes the model unification and constraint analysis, can be performed between the CIM and PIM levels of MDA. This practical implementation moves TEAEM from theory to application, allowing us to test it against real-world-inspired scenarios—such as our running example (medical supply chain traceability) case where trust, compliance, and technical feasibility must be balanced.

The implementation is structured into three modules, each playing a distinct role in moving from separate business and technical models to a unified, analyzable system model[1]:

1. **Generating the Unified Feature Model:** For goal modeling, we use the istar (*) language (Franch et al., 2016), and PiStar tool (Pimentel and Castro, 2018) to represent the business goals, dependencies, and high-level trust requirements. For feature modeling, we use FeatureIDE (Kastner et al., 2009), which captures abstract technical functionalities. The mapping between the two levels—linking business goals to technical features is defined manually in an external file using logical expressions. A dedicated Python script, transform.py, reads the goal model, the feature model, and the constraint mappings, and automatically transforms them into a single unified model that captures all dependencies and trade-offs.

2. **Extraction of the constraints from the UFM:** We use FeatureIDE and its bundled Sat4j solver Le Berre and Parrain, 2010 to retrieve the constraints from the UFM. We generate an output in JSON that is further used for the constraint analysis step.

3. **Interpretation of the solution** *constraints violation and their implication bottom-up*: We develop a Python module [interpretation.py] that uses the .json list of constraint violations extracted from the UFM in FeatureIDE as an input and produces the domain-specific interpretations of these violations that can be further used for enterprise models redesign.

For example, in the medical traceability scenario, the solver might detect that enabling "full blockchain transparency" conflicts with the goal "Protect sensitive manufacturing data". The interpretation module would then translate this into a plain-language explanation:

> *"This configuration increases auditability but compromises confidentiality. Consider using selective data encryption to balance both goals.*

This translation step is crucial: it bridges the gap between technical analysis and actionable business decisions, allowing enterprise architects and decision-makers to redesign models with a clear understanding of the trade-offs involved.

6.2.3 Illustrative Use Case

This use case presents a practical example demonstrating the application of the TEAEM Approach. The scenario involves Organizations that seeks to enhance transparency and

[1]https://github.com/Eddykams/TEAEM_develop

auditability of its processes by using blockchain technology as a platform for its enterprise solutions. In particular, the Organization needs to choose between two blockchain platforms Public Blockchain (PublicBC_X), and Private blockchain (PrivateBC_Y). Further, in the design, it will configure the selected blockchain platform and its components. Since the Organization is also concerned with the GDPR compliance, more detailed analysis of the abstract capabilities and technical functionality of the prospective blockchain solution is required. We illustrate how TEAEM can support the analysis of technical constraints and their impact on the defined business goals of the organization.

Step 1: Goal Modeling (CIM): The organization's high-level business objectives are modeled. The goal model illustrated in Fig. 6.2.(a) defines the two goals of the Organization: 'GDPR Compliance' and 'Auditability' that have to be achieved. Their satisfaction is expressed through an i* AND-refinement, indicating that both must hold for the overarching business objective to be achieved. The logical conjunction:

```
Business Goals => GDPR Compliance AND Auditability
```

Step 2: Feature Modeling (PIM): The abstract technical capabilities of prospective blockchain solutions are modeled in the feature model illustrated in Fig 6.2.(b). Here, the organization must choose between the two alternatives platforms: Public or Private blockchain. This exclusive choice is represented as a logical XOR (exclusive OR) in the feature model (PublicBC_X XOR PrivateBC_Y).

```
Technical Solution => PublicBC_X XOR PrivateBC_Y
```

Step 3: Component Modeling: On the first version of the implementation of TEAEM, we do not take account the integration of the component modeling (PSM). We limited the tool with the mapping of CIM-PIM to create a unified model (UFM).

Once the goal model and solution proposition are formalized, we proceed with the (bottom-up) constraint propagation and analysis.

Step 4: Unified Feature Model (UFM) We create a unified feature model that specifies the 'Enterprise System' of the Organization (Fig. 6.2.(c)). Our UFM represents the goals (from CIM) and the features (from PIM) using the feature modeling formalism. Here we model 'Technical Solution' and 'Business Goals' as mandatory features of the 'Enterprise System' root feature.

To formalize the mapping between the created MDA models, We assume the following assumptions: we consider generic properties of Publicblockchain X and Private blockchain Y. The Private blockchain Y limits the auditability of the blockchain because

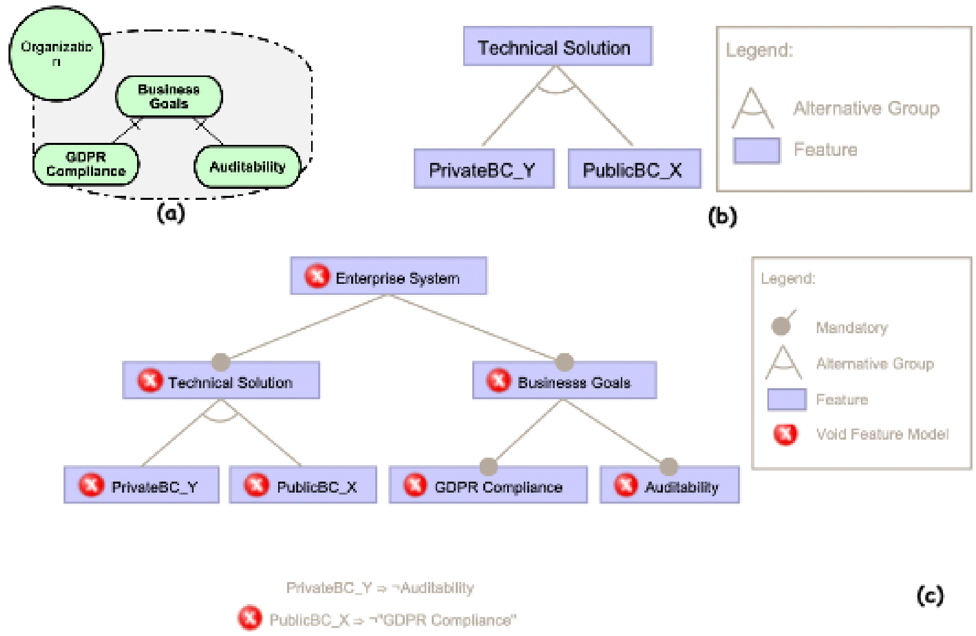

FIGURE 6.2: (a): Goal Model of the Organization; (b): Feature Model with a proposition of alternative blockchain solutions; (c): Unified Feature Model that merges (a) and (b).

their nodes are controlled by a single (private) entity and the immutability is contingent upon the trustworthiness of this entity.

The Public blockchain X does not face this immutability issue because any node can participate in the consensus and validation processes, ensuring that operations are conducted fairly and transparently. Nevertheless, this immutability of public blockchains prevents from implementation of a fundamental GDPR's 'right to be forgotten' - an alteration/deletion of data on demand. We map the technical features defined in PIM to the goals that can be compromised by implementation of these features using the following logical expressions:

```
PublicBC_X => NOT GDPR Compliance;
PrivateBC_Y => NOT Audit
```

Step 5: Model checking/ Impact analysis: The FeatureIDE automatically executes the model-checking on the generated UFM. In our example, the UFM is invalid (i.e., the model is `void`), meaning that no Enterprise System satisfying actual constraints can be configured. We identify and trace the sources of conflicting constraints to the model elements defined at different MDA levels (PIM and CIM in our example). Using our developed Extraction and Solution Interpretation modules, we provide the domain-specific constraint interpretation, aiming to assist business users in redesign.

```
Logical constraints:
 ['Enterprise System ⇒ Technical Solution', 'Enterprise System ⇒ Businesss Goals', 'Techni
cal Solution ⇒ PrivateBC_Y v PublicBC_X', 'Businesss Goals ⇒ GDPR Compliance', 'Businesss
Goals ⇒ Auditability']
*********************************

Domain-specific constraint analysis & interpretation:
--------------------------------------------------------
To correct your UFM, you have the following options:
-------------------------------------------------------
Make constraint 'GDPR Compliance' non-mandatory, OR Remove constraint 'GDPR Compliance'.
Make constraint 'Auditability' non-mandatory, OR Remove constraint 'Auditability'.
```

FIGURE 6.3: Domain-specific constraint analysis & interpretation.

In this example, the business expert faces the problem where any choice of blockchain solution compromises one of the business goals defined by the organization. Along with modification or refinement of the prospective Technical Solution, this analysis calls for redefinition or re-prioritization of the Business Goals in response of the bottom-up constraint propagation.

6.2.4 Discussion

The illustrative use case provides a preliminary validation of the first of TEAEM and highlights its current limitations. The use case provides concrete evidence that the core TEAEM approach is viable and valuable. It successfully demonstrates the framework capacity to:

1. *Unify Models:* TEAEM operationally unify models originating from the MDA levels namely, goal models at the CIM level and feature models at the PIM level—into a single coherent representation (the Unified Feature Model). This integration enables technical constraints defined at lower levels to propagate upward toward business objectives.

2. *Propagate and Reveal Latent Conflicts:* The framework's primary contribution, bottom-up constraint propagation, is validated. The model-checking reveals a structural inconsistency in the UFM: neither PublicBC_X nor PrivateBC_Y can simultaneously satisfy both GDPR compliance and auditability. This configuration exemplifies TEAEM's capacity to detect incompatibilities that emerge from the interplay between technical design choices and high-level organizational goals. By tracing these conflicts back to their originating model elements, the framework provides domain-specific interpretation that can guide business experts in reconsidering goal definitions or exploring alternative technical solutions.

3. *Support Informed Redesign:* By automatically detecting the conflict and tracing its source to specific model elements, TEAEM moves beyond mere identification

to provide actionable insight. The domain-specific interpretation of constraints (e.g., explaining why a public blockchain violates GDPR's "right to be forgotten") shifts the output from a technical error log to a business-reasoning aid, directly informing potential redesigns of either the technical solution or the business goals.

6.2.5 Limitations

The first version of TEAEM provided as structured way to unify into a single model business goals, technical features and implementation components, and to analyze how design decisions at one level might affect higher levels of abstraction. However, this initial implementation came with important limitations.

A first limitation is that TEAEM (v1) did not explicitly integrate soft goals (as dimension of trust). While hard goals could be represented and mapped to features and components, qualitative trade-offs, such as balancing transparency with privacy—were only indirectly captured. This meant that conflicts between competing soft goals were not reasoned about in the first version of the framework.

The second limitation concerned automation. The first version was Semi-automated. Although the generation of the Unified Feature Model and constraint checking were automated, the mapping between goals and features still had to be provided manually with external file. This increased the effort required from designers and introduced the risk of human error.

6.3 TEAEM-Extension: Why TEAEM Matters for Trust

In practice, many (trust) requirements are soft goals. For example, in our running example, goals can be "maximize transparency", "minimize cost", or "protect confidentiality". These are qualitative, not absolute, and they often interact in complex ways (e.g: increasing transparency for regulators might reduce confidentiality for manufacturers.)

TEAEM make these soft goals explicit at the CIM and PIM levels, so they can be systematically evaluated against technical options at the PSM level. In our case, selecting full ledger transparency would score high for auditability but low for data confidentiality. By making these trade-offs visible early, TEAEM enables stakeholders to adjust priorities before committing to an implementation path.

The limitations of TEAEM(v1) highlighted two main gaps: the absence of support for the Platform-Specific Model (PSM) and the lack of mechanisms to incorporate soft goals into the configuration process. In practice, this restricted the ability of business users and system engineers to reason about qualitative attributes such as cost, security, and performance.

To overcome these limitations, we introduce TEAEM Extension (TEAEM-Ext). This version provides three main enhancements:

- **Integration of the PSM:** Technological decisions are captured through SysML component modeling, ensuring that platform-level design choices are reflected consistently across all abstraction layers.
- **Explicit modeling of soft goals:** Unlike TEAEM(v1), which focused mainly on functional goals, TEAEM Extension incorporates qualitative attributes into the modeling process. These soft goals considered as dimension of trust are quantified through utility functions to support systematic evaluation.
- **Configuration generation guided by soft goals:** Using Multiple-Criteria Decision-Making (MCDM) methods, such as TOPSIS, TEAEM Extension generates and ranks alternative configurations, enabling business users to select the one that best aligns with their preferences.

These extensions strengthen TEAEM by combining top-down design with bottom-up traceability, while enabling the evaluation and optimization of soft goals during product configuration. The following sections detail the conceptual foundations of TEAEM Extension and illustrate its application to Software Product Lines.

6.3.1 Software Product Lines

MDA and SPL are complementary approaches that greatly benefit from the integration of soft goals. Software Product Line Engineering (SPLE) serves as a methodological framework for developing software families, characterized by significant overlap in functional and non-functional attributes (Bošković et al., 2011).

A primary focus of SPLE is to identify and codify the common features of a cluster of software-intensive applications within a specific domain (Bagheri et al., 2011). An SPL is generally composed of three main types of artifacts:

- *Problem Space*: this describes the features and functionalities desired by users across all potential products in the family.
- *Solution Space:* This includes the design and implementation specifics for all components across different products within the SPL.
- *The Mappings:* These establish the connections between features in the problem space and their respective components in the solution space.

In this work, we choose SPL like application domain of TEAEM.

6.3.2 Soft goals and Utility Functions for Evaluating Soft goals

Soft goals serve as quality criteria that software systems or their components must meet during development. They are subjective, relative, diverse, and interactive, making them crucial yet challenging to model due to their qualitative nature (Hu et al., 2015).

The literature highlights the significance of soft goals in determining the success of software projects. For instance, consider the soft goals for a software system such as *performance* ("The system should respond quickly to user inputs") and *security* ("The system should protect user data from unauthorized access"). These qualitative goals pose the question: how can we effectively evaluate them?

Significant research has been conducted on identifying soft goals within Information Systems (IS). For example, one study presented a comprehensive list of 114 soft goals identified in information Systems (Mairiza et al., 2010), while another proposed a taxonomy categorizing these soft goals into four groups: Real and Web-based Systems, Web-based Systems, Real-time Systems, and Information Systems (Gazi et al., 2015). We aim to adapt some of these identified soft goals and quantify them.

To quantify soft goals, we propose writing a utility function for each soft goal, facilitating the evaluation of how well a system meets these goals. Soft goals like usability, reliability, and performance are inherently qualitative. By converting these into quantifiable metrics through a utility function, we enable more objective assessments and comparisons.

6.3.3 TEAEM Extension: Design and Modeling Principles

Building on the conceptual improvements introduced above, TEAEM Extension operationalizes these ideas through a unified modeling process. Figure 6.4 illustrates the overall workflow, which combines the top-down MDA design layers (CIM, PIM, PSM) with bottom-up validation, constraint propagation, and configuration generation guided by soft goals.

- Integration of PSM to illustrate technological choices using component modeling (SysML).
- Integration of soft goals to guide business users in generating configurations within the SPL.

This integration ensures that technological choices made at the PSM level are consistently reflected across all abstraction levels, enhancing the overall coherence and bottom-up traceability within the enterprise modeling process. Second, we extend TEAEM by integrating soft goals into the CIM. By including soft goals in the early stages of modeling, we can better address qualitative attributes throughout the system development life-cycle. This approach allows for the generation of code (configuration) guided by the satisfaction of these soft goals, thereby optimizing the system for user satisfaction and strategic business goals.

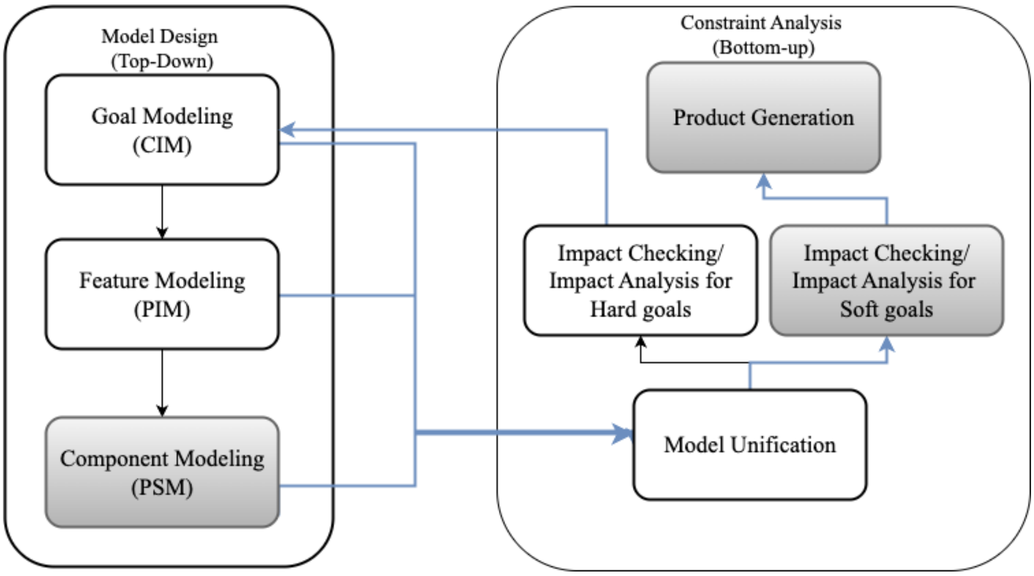

FIGURE 6.4: TEAEM approach to meet Soft goals. Hashed blocks show TEAEM extension proposed in this paper

On the left-hand side represents the top-down MDA design approach with three abstraction levels: CIM, PIM, and PSM.

- *CIM Layer:* We start by creating the goal model using the i* frameworkFranch et al., 2016 with the PiStar tool(Pimentel and Castro, 2018). This model includes both hard goals (functional requirements) and soft goals (non-functional requirements).
- PIM Layer: We design the Feature Model using the FeatureIDE tool (Kastner et al., 2009).
- PSM Layer: We create component models with SysML, which include technical metadata about alternative implementations of the designed solution.

On the right-hand side, we extend TEAEM by unifying the PIM, CIM, and PSM layers of MDA into a single model called the "Unified Feature Model (UFM)". This unification is crucial for several reasons:

- Model checking and validation help validate the UFM to confirm it meets requirements, identify inconsistencies in the specification, and show conflicts between elements of the unified feature models.
- Impact Analysis: Helps identify how changes in one part of the model affect others by reporting constraint violations from low-level to high-level.
- Configuration Generation in SPL: Facilitates the generation of configurations that satisfy soft goals.

To achieve this unification, we define the following mapping:

- *Goal to Feature Mapping* (TEAEM): This mapping associates business goals with the technical features that fulfill those goals. It reflects the technical assumptions made by system engineers.
- *Feature to Goal Mapping* (TEAEM): This mapping associates technical features with the goals that could potentially be compromised or hindered by their implementation.
- *Feature to Component Mapping* (TEAEM Extension): This new mapping associates features with the specific technological components that implement them.

The mappings are formalized using a symbolic mathematics library to structure the logical expressions. This library also develops utility functions to optimize the soft goals.

After unifying the model, we use model-checking with a boolean solver to validate the UFM and identify inconsistencies causing conflicts between goals, features, and components.
If model checking returns false, indicating no feasible configuration, we provide a domain-specific interpretation of these conflicts and recommend solutions. If model checking returns true, indicating multiple feasible configurations, we integrate soft goals into the decision-making process. We use a utility function to quantify each soft goal and apply a Multiple-Criteria Decision-Making (MCDM) algorithm to optimize soft goal satisfaction, determining the best configuration for the business user.

The main distinction between TEAEM and other frameworks like MDA or ArchiMate is its integration and analysis of inconsistencies across different abstract level (goal, feature, component) within the MDA. While ArchiMate offers a holistic view of enterprise architectures, it does not specifically focus on detecting and managing inconsistencies between different model types or guiding product generation by soft goals.

6.3.4 Modeling Soft goals with the Utility Function

As discussed in Section 6.3.3, TEAEM-Ext integrates soft goals (dimensions of trust) directly into the Unified Feature Model (UFM). To operationalize this, each soft goal is associated with a utility function that quantifies how well a given configuration satisfies the corresponding qualitative attribute.

Soft goals are inherently subjective (e.g., cost, response time, throughput), making them difficult to evaluate through traditional Key Performance Indicators (KPIs). While KPIs offer isolated measurements, they do not capture trade-offs across competing criteria.

Utility functions, in contrast, provide a single quantitative measure that integrates multiple dimensions, enabling systematic comparison of configurations.

The utility function quantifies values derived from component modeling (value of each component) and serves as input for the MCDM algorithm to generate configurations that maximize user satisfaction. We focus on the following sof tgoals

Table 6.1 summarizes the soft goals selected for our implementation and the component properties they are refined into.

Soft goal	**Description**	**Refined into**	**Notation**
Cost	System should be cost-efficient.	Deployment and maintenance expenses.	C
Response Time	System should be fast.	Transaction, block creation, and contract execution times.	R
Throughput	System should support high throughput.	Block size, block time, and network bandwidth.	T

TABLE 6.1: Description of soft goals and their refinement in blockchain systems

The global objective is to optimize configurations by minimizing cost (C) and response time (R) while maximizing throughput (T). This is expressed as:

$$\min F = w_1 C + w_2 R - w_3 T$$

Where F is the combined objective function, and w_1, w_2, w_3 are weights representing the relative importance of each soft goal according to business user preferences.

This formulation provides a direct input to the TOPSIS algorithm during configuration selection, enabling TEAEM-Ext to balance trade-offs between competing soft goals in a systematic and transparent manner.

6.3.5 TEAEM-Extension in Practice: Implementation

We implemented TEAEM Extension as an automated process that unifies models across MDA abstraction levels, validates consistency, performs impact analysis, and generates configurations guided by soft goals. The implementation is available as open source[2].

The workflow is organized into the following modules:

[2]https://github.com/Eddykams/TEAEM_develop

1. **Unified Feature Model (UFM) Generation:** This module consolidates the goal model (CIM), feature model (PIM), and component models (PSM) into a single UFM. Goals are modeled with PiStar, features with FeatureIDE, and components with SysML. Logical mappings across levels are expressed with SymPy to ensure automation and formal consistency.

2. **Constraint Extraction and Validation:** Using the Sat4j solver embedded in FeatureIDE Le Berre and Parrain, 2010, constraints are extracted from the UFM and checked for satisfiability. Inconsistent specifications are reported as JSON files for further analysis.

3. **Interpretation of Violations:** When inconsistencies occur, the system interprets the solver's output in domain-specific terms (e.g., conflicting goals, incompatible components), providing business users with actionable feedback for refinement.

4. **Soft goal Evaluation with Utility Functions:** Component metadata from the SysML models (e.g., cost, block size, transaction times) are used as parameters for the utility functions introduced in Section 6.3.4. These functions quantify soft goal satisfaction for each candidate configuration.

5. **Optimization with TOPSIS:** Multiple-Criteria Decision-Making (MCDM) is performed using the TOPSIS method. By combining utility scores with user-defined weights, this module ranks candidate configurations and identifies the one(s) best aligned with business user preferences.

This modular implementation ensures separation of concerns: unification, validation, interpretation, and optimization are handled independently, yet integrated into a single workflow. The result is an automated pipeline that allows business users to generate and evaluate SPL configurations with explicit consideration of both hard goals and soft goals.

6.4 Configuration of Software Product Lines Driven by the Soft goals: the TEAEM Approach (Running Example)

6.4.1 Overview of the Use Case

The market of counterfeit drugs has become a 200-billion-dollar business annually, according to the World Health Organization (WHO). According to a WHO report, up to 10% of all sold drugs globally are fake, with significantly higher rates in parts of Africa and Asia. This business is very dangerous because of life-and-death implications for patients (thousands of deaths every year) and also for the pharmaceutical industry's reputation, which can lose a lot of money. The complexity of pharmaceutical supply chain operations is the primary reason for this issue. With many handovers to different supply chain partners before drugs reach hospitals and pharmacies, there is a significant lack of traceability. Manufacturing is vulnerable to counterfeit raw materials or ingredients from unknown sources. Moreover, illicit producers can relabel fake products to infiltrate legitimate distribution channels. For the pharmaceutical sector, the European Union (EU) and the Drug Supply Chain Security Act (DSCSA) in the United States provide a deadline of 2023 for the industry to implement a traceability system.

Based on the work (Kambilo et al., 2022; Hastig and Sodhi, 2020), we present the following illustrative example drawn in Chapter 3. The organization has many branches in different countries that require a system to manage transactions recorded across multiple locations. Additionally, there is a need for a system that enhances trust among various stakeholders.

The system must address the following needs:

- *Reduce drug counterfeiting (fraud detection)* by ensuring traceability and transparency at every step of the product transfer.
- Ensure *secure data management* by enhancing privacy through access controls and encryption, and by maintaining the integrity and immutability of data.
- *Manage recalls of defective products*; conditions of transporting drugs must be verified through real-time audits using IoT, sensors, or RFID tags. The authenticity of products must be checked by log-audit.

6.4.2 Applying and Validating TEAEM-Ext

The main goal of the organization is to determine if, with their goals, features, and components, they can generate a product or if there are any inconsistencies between them before generating the product. The organization proposes to integrate the following soft goals: ***maximal performance, minimal cost for blockchain components, better response time, and minimal cost for the electronic solutions (RFIDs, IoT sensors, or NFC tags)***.

We demonstrate how TEAEM-Ext supports the configuration of Software Product Lines (SPLs) to meet soft goals in this context. The process unfolds in six steps:

Goal Modeling: A partial goal model (Figure 6.5) is created using the i* framework, capturing both hard goals (e.g., "Fraud Detection") and soft goals (e.g., "Minimal Cost", "High Performance").

Feature Modeling To build the feature model, we made a literature review to identify different solutions proposed for combating drug counterfeiting in the supply chain illustrated in Table 6.2. According to some findings in the literature, (Bapatla et al., 2023; Rajora, 2023) assert, for example, that the ideal anti-counterfeit technology in an enterprise system should have a high level of safety, faster product application, established standards, simple to check, have automatic authentication, be accessible to consumers, and comply with industry regulations.

We built the feature model based on the technologies identified in Table6.2. Three prominent technologies were proposed: blockchain technology, for its features such as traceability, trust, and immutability; Cloud Computing (Vijayaraj et al., 2024), known for its flexibility and faster application production; and Machine Learning, which can identify counterfeit drugs through algorithmic data analysis. Additionally, associated electronic equipment for traceability is proposed, such as RFID tags, IoT sensors (pressure, geolocation), and NFC tags.

Goal / Requirement	Blockchain	Cloud	Machine Learning
Transparency	✓	–	–
Traceability	✓	–	✓
Reliability	✓	✓	✓
High Level of Safety	✓	–	–
Fast Product Application	–	✓	–
Ease of Auditing	✓	–	–
Automated Processes	✓	✓	–
Regulatory Compliance	–	✓	✓
Accountability	–	✓	–
Trust	✓	–	–
Decentralization	✓	–	–
High Security	✓	–	–
Flexibility	–	✓	–

TABLE 6.2: Mapping of Requirements to Enabling Technologies

Component Modeling: Based on the analysis presented in Table 6.2, we decided to use blockchain technology as a component to illustrate our example. Blockchain is a novel technology that has not yet been widely adopted by enterprises due to the lack of standardization and scarcity of engineers skilled in its implementation. The literature identifies two principal types of blockchains: public blockchain and private

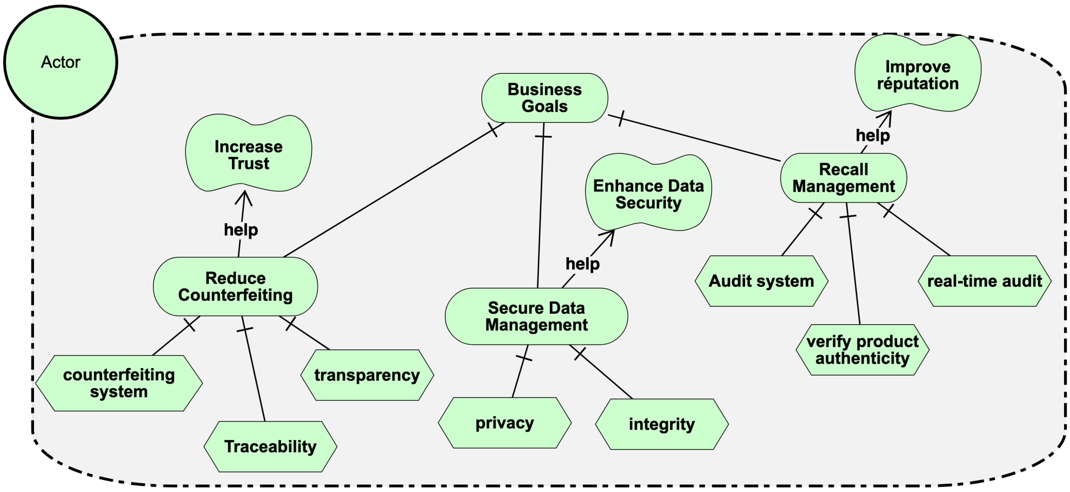

FIGURE 6.5: Goal Modeling of Enterprise Safri-Import

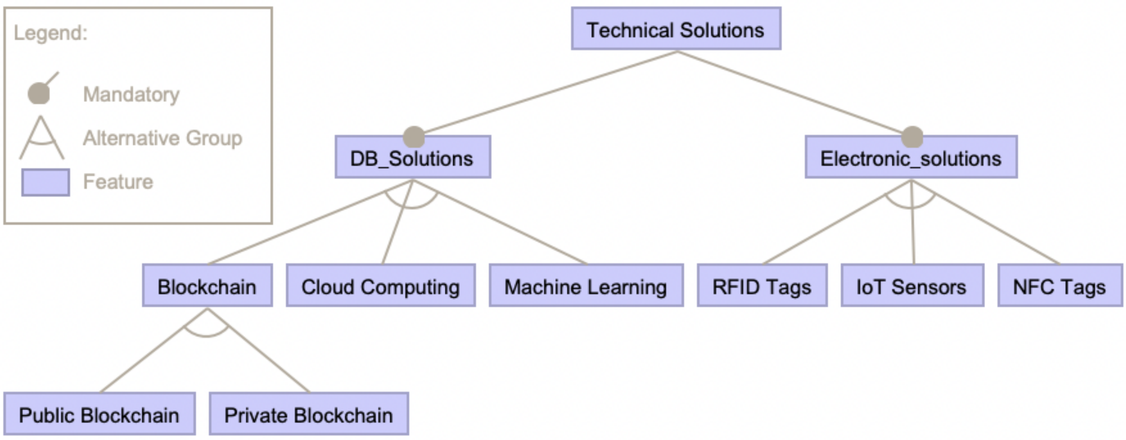

FIGURE 6.6: Feature Model of Enterprise Safri-Import

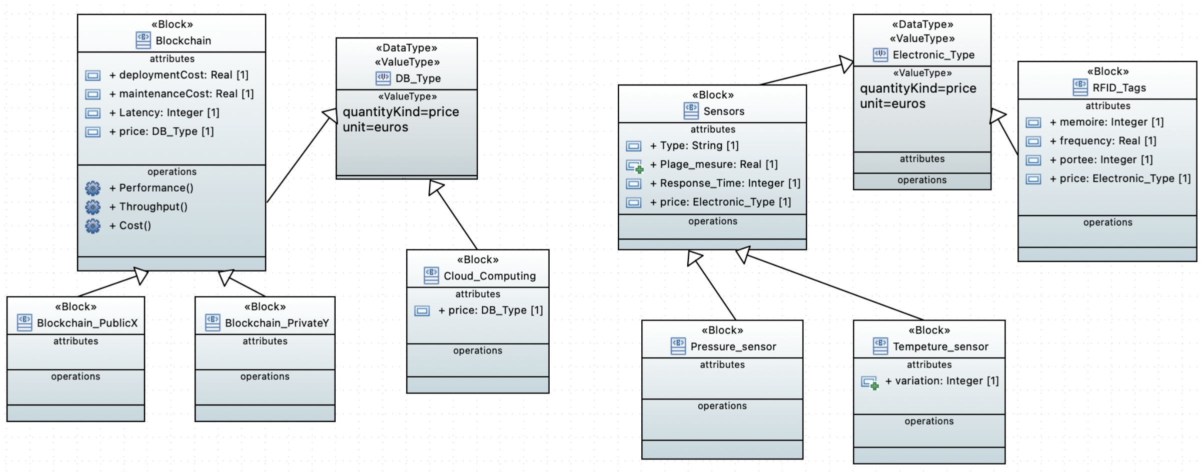

FIGURE 6.7: Component Model of Enterprise Safri-Import

blockchain. Using SysML, we illustrate the component model for blockchain technology and other technologies. However, for the evaluation phase, we focus on the value of the blockchain component.

Model Unification: The unified model, as depicted in Fig. 6.8, put within a single model the three abstract level of the MDA. In our study, the model checking results were positive, indicating that the unified model is valid. However, the next step involves generating configurations that satisfy the soft goals defined in the previous section.

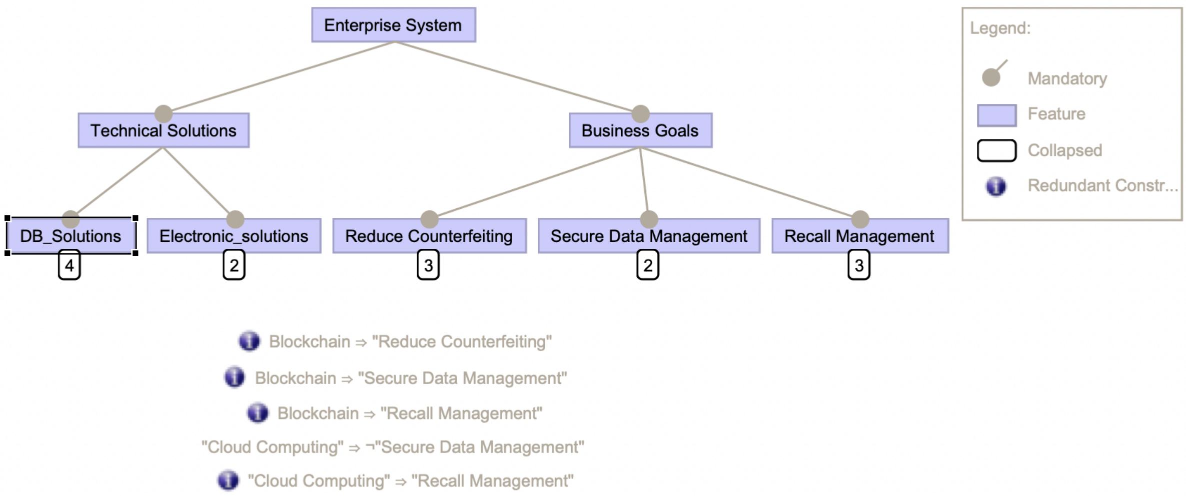

FIGURE 6.8: Model Unification

Model Checking/Impact Analysis: After model unification, we use model checking/validation to determine if a valid configuration exists for this unified model or if there are any inconsistencies. *The unified feature model generates 4 possible configurations*, indicating that there are no inconsistencies or constraint violations. Therefore, refining the model is unnecessary. In 6.2.2, we presented an example where constraint violations were detected. The question then arises: how do business users choose the "best" configuration that meets their soft goals?

Optimization with TOPSIS algorithm: As outlined in the previous section, to generate configurations guided by the satisfaction of soft goals, we detail the soft goals discussed in subsection 6.3.4, including their associated utility functions and the blockchain components employed. The values of component properties are inputted as parameters into the utility functions to select the optimal configuration. We use blockchain configurations and utilize the TOPSIS method for multi-objective optimization.

In the absence of a generalizable throughput/response time formula for all possible cases, we have chosen to formulate the following assumptions: Response Time is the minimal time for the detection of counterfeiting drugs and depends on the component properties: Transaction Processing Time, Block Creation Time, and Smart Contract Execution Time. The throughput is a function of block size, block execution time, and network bandwidth for the blockchain.

Soft Goal	**Utility Function**	**Component Properties**
Blockchain Cost	$C = \min \sum_{i=1}^{n} C_i(D_c, M_c)$	Deployment cost Maintenance cost
Throughput	$T = T(B_s, B_t, N_b)$	Block size Block time Network bandwidth
Response Time	$R = R(Tx_i, B_c, SC_e)$	Transaction processing time Block creation time Smart contract execution
Electronic Cost	$E_C = \min \sum_{i=1}^{n} C_i(D_c, E_c, M_c)$	Deployment cost Execution cost Maintenance cost

TABLE 6.3: Utility functions and blockchain component properties of soft goals

6.4.3 Results

After integrating the blockchain component values as parameters in the utility function to generate products, we made the following tests:

Guiding Configurations for a Single Soft Goal

When an organization prioritizes a single objective, such as minimizing blockchain costs, the system successfully identifies the optimal configuration. The implemented module maps the component value across configurations, calculates its utility via a SymPy function, and selects the configuration with the minimal cost.

Guiding Configurations for Multiple Soft Goals

Real-world scenarios often require balancing multiple, often conflicting, criteria, leading to a multi-objective optimization problem. We implement the TOPSIS method for Multiple-Criteria Decision-Making (MCDM) to handle this.
These values do not result from the utility function; they serve solely to demonstrate the application of MCDM using TOPSIS. The four configurations correspond to those

```
 *********** Components Mapping Completed. ***************
Analyze of configuration: conf1.xml
Selected Feature: PublicBC_X
PublicBC_X - Ethereum: Price = 40.0
###
Analyze of configuration: conf0.xml
Selected Feature: PrivateBC_Y
PrivateBC_Y - Hyperledger: Price = 35.0
 ************ DECISION GUIDING CHOICE ********************
The utility function: 35.0
To reduce Cost, The best Configuration is: conf0.xml
 ******** ************ **********************
```

FIGURE 6.9: Configuration guiding by the Minimal Blockchain Cost

Attributes	Blockchain Cost	Response_Time	Throughput	Electronic Cost
Conf0	300	15 seconds/bloc	18 Tx/seconds	140
Conf1	350	15 seconds/bloc	16 Tx/seconds	120
Conf2	200	2 seconds/bloc	100 Tx/seconds	200
Conf3	250	3 seconds/bloc	200 Tx/seconds	180

TABLE 6.4: Soft goals Criteria with Height

identified after model checking, and their associated values were provided as input to evaluate trade-offs across criteria.

The table 6.4 illustrates the four configurations found after the model checking/validation. Different values (Blockchain Cost, Response Time, Throughput, and Electronic Cost) were passed as inputs to our component model. In this case, we have a multi-objective optimization where we need to choose the best configuration guided by multiple soft goal criteria. Many optimization algorithms exist in the literature, and we implemented the TOPSIS method for Multiple-Criteria Decision-Making.

6.5 Evaluation

The validation phase consists in applying the TOPSIS method to the data in Table 6.4. Two scenarios were considered to assess how variations in decision-maker preferences influence the recommended configuration.

6.5.1 Scenario 1: Equal Weights

We defined the same weight for all parameters [0.25,0.25,0.25,0.25] and identified Response Time, Blockchain Cost, and Electronic Cost as non-benefit parameters (functions to minimize) and Throughput as a benefit parameter (function to maximize). We have the following outcomes:

Attributes	BC_Cost	R_Time	Throughput	E_Cost	S_POS	S_NEG	Score	Rank
Conf0	300	15	18	140	0.2215	0.0862	0.2801	3
Conf1	350	15	16	120	0.2349	0.0613	0.2069	4
Conf2	200	2	100	200	0.0689	0.2139	0.7561	2
Conf3	250	3	200	180	0.0532	0.2215	0.8061	1

TABLE 6.5: Scenario 1 Outcomes Overview

6.5.2 Scenario 2: Priority on Response Time

Business users place significant emphasis on Response_Time compared to other criteria. We adjust the weight according to preferences, for example, 0.6 for Response Time and 0.13 Blockchain Cost, 0.13 Throughput and 0.13 Electronic Cost ([0.13, 0.61, 0.13, 0.13]).

Attributes	BC_Cost	R_Time	Throughput	E_Cost	S_POS	S_NEG	Score	Rank
Conf0	300	15	18	140	0.3682	0.0344	0.0856	3
Conf1	350	15	16	120	0.3695	0.0245	0.0622	4
Conf2	200	2	100	200	0.0275	0.3675	0.9301	1
Conf3	250	3	200	180	0.0347	0.3416	0.9075	2

TABLE 6.6: Outcomes for Scenario 2

6.5.3 Results

For the first scenario, to reduce the blockchain cost, the cost of the electronic equipment, and the response time, while increasing performance with throughput, the best configuration is Configuration 3 (Conf3), which achieves 80% satisfaction among business users. The worst configuration is Configuration 1 (Conf1), with only 20% satisfaction among business users. This is clearly specified with different constraints and the values of blockchain components as parameters. The configuration that meets the specified soft goals is Configuration 3.

For the second scenario, the best configuration has changed and is now Configuration 2 (Conf2), with 92% satisfaction among business users. The worst configuration is still Configuration 1 (Conf1), with only 6% satisfaction among business users. We can clearly see that a different soft goal weight will lead to different configurations.

This example demonstrates how TEAEM Extension systematically guides configuration in Software Product Lines (SPLs). By combining top-down modeling with bottom-up evaluation, the approach:

- Detects inconsistencies early through model checking.
- Quantifies trade-offs between cost, performance, and other soft goals.
- Provides ranked configuration alternatives tailored to business user preferences.

6.5.4 Discussion

The TEAEM approach by integrating soft goals ensures that qualitative attributes such as user satisfaction, security, and performance are prioritized, resulting in configurations that align more closely with business goals. The unified CIM-PIM-PSM model enhances coherence and traceability across abstraction levels, leading to fewer errors and more complete configurations. This integration ensures that technological choices made at the PSM level are consistently reflected across all levels, reducing the risk of inconsistencies. TEAEM generated configurations exhibit fewer errors and higher completeness, as validated by model checking.

The use of the TOPSIS method for MCDM in TEAEM facilitates the selection of optimal configurations based on multiple soft goal criteria. This approach allows business users to make informed decisions quickly, balancing trade-offs between competing goals. Empirical data show that TEAEM reduces the time required for configuration generation and decision-making compared to manual methods and other SPL approaches.

Compared to other approaches, TEAEM adds value in model-checking validation and constraints analysis propagation for bottom-up traceability. MDA focuses on top-down design, traceability, and alignment of goals from business goals to code generation. ArchiMate is designed to provide a comprehensive, integrated view of enterprise architecture, focusing on the relationships between different domains (such as business, application, and technology). However, it does not inherently include specific mechanisms for detecting and managing inconsistencies between different model types. Additionally, while ArchiMate can represent various goals and requirements, it does not explicitly focus on guiding product generation through soft goals.

The implementation of TEAEM and the various evaluations carried out have allowed us to learn the following lesson:

- *Business users have the ability to express their preferences on the different soft goals, and expressing these preferences will have a technical impact. The weight or importance that a business user assigns to a soft goal during the configuration of a product in a Software Product Line (SPL) significantly influences the best configuration that can be proposed to them.*

6.6 Threats to Validity

Although the proposed TEAEM Extension approach demonstrates promising results, several threats to validity must be acknowledged.

6.6.1 Internal Validity

The illustrative example was based on assumptions regarding component properties (e.g., block size, processing time, costs). These assumptions may not fully reflect real-world conditions and could bias the outcomes of the optimization. Furthermore, the evaluation relied on the correctness of the implemented utility functions. Inaccuracies in their formulation would directly affect the ranking of configurations.

6.6.2 External Validity

The evaluation was conducted in a single application domain (counterfeit drug detection in the pharmaceutical supply chain). While this domain is representative of trust-critical scenarios, the generalizability of the results to other domains (e.g., finance, logistics, e-government) has not yet been empirically validated.

6.6.3 Construct Validity

The evaluation was conducted in a single application domain (counterfeit drug detection in the pharmaceutical supply chain). While this domain is representative of trust-critical scenarios, the generalizability of the results to other domains (e.g., finance, logistics, e-government) has not yet been empirically validated.

6.6.4 Conclusion Validity

The evaluation was conducted in a single application domain (counterfeit drug detection in the pharmaceutical supply chain). While this domain is representative of trust-critical scenarios, the generalizability of the results to other domains (e.g., finance, logistics, e-government) has not yet been empirically validated.

Recognizing these threats helps delimit the scope of our current contributions and provides guidance for future empirical work.

6.7 Conclusion

In this chapter, we presented TEAEM approach that contributes to design of enterprise systems within the MDA paradigm. While existing approaches support traceability and alignment between goals and technical solutions top-down, they often fall short in guiding stakeholders through the decision-making process and analysis of inconsistencies bottom-up. Our approach enables reconciliation between business goals, technical capabilities and specific solutions through formal analysis and propagation of constraints.

Our primary contribution is the unification of the CIM-PIM-PSM into a single model, ensuring a traceable design process that connects business goals with technical implementations. We also integrate model checking and impact analysis to detect inconsistencies early, aiding decision-making and minimizing errors. The second contribution integrates soft goals as dimension of trust into the TEAEM process, enabling SPL configurations to meet both goals and soft goals, ensuring the final products align with both user needs and business goals.

The feasibility of the proposed approach was illustrated through examples drawn from the supply chain domain, including a counterfeit drug detection scenario leveraging blockchain technology. These examples demonstrated how technical properties of specific solutions can be propagated to higher-level organizational models and how stakeholder preferences can guide configuration choices in the presence of competing goals and soft goals. While some illustrations remain conceptual, they provide initial evidence of the potential of TEAEM to support informed and transparent design decisions in inter-organizational settings.

While the results are promising, the work also revealed several challenges. The formalization of utility functions for complex soft goals, the scalability of feature models, and the generalizability of the approach across domains remain open issues. These challenges define an agenda for future research. In particular, we envision extending TEAEM-Ext with more expressive modeling of non-linear utility functions, exploring modularity techniques to reduce feature model complexity, and conducting empirical evaluations in diverse industrial domains.

Overall, this chapter positions TEAEM as a step toward more informed and trust-aware enterprise system design, by bridging business goals, qualitative concerns, and technological choices within a unified model-driven framework. In doing so, it lays the foundation for the subsequent chapters, which focus on the operationalization and evaluation of trust-related concerns in blockchain-enabled inter-organizational business processes.

Chapter 7

BC-TEAEM Knowledge Base and Recommendation Engine

This chapter present BC-TEAEM Knowledge Base, an ontology-driven framework that formalizes blockchain features and patterns, and trust-related soft goals together with their interdependencies. The proposed knowledge base provides a structured representation of how blockchain patterns influence key trust dimensions such as performance, security, and transparency, and how these dimensions interact through neutral, positive and negative dependencies. By integrating this formal knowledge-base with the TEAEM approach and a multi-criteria analysis mechanism, the framework supports systematic trade-off analysis, preference elicitation, and reasoned pattern selection to meet trust. The chapter further demonstrates how this knowledge base can be operationalized to guide configuration generation and design decisions in trust-sensitive inter-organizational contexts, with a particular focus on supply chain management.

7.1 Introduction

Blockchain technology is increasingly adopted in inter-organizational and distributed systems to ensure trust, transparency, traceability, and immutability. Its integration is particularly relevant in domains where collaboration occurs without full mutual trust, such as supply chain management.

The design of blockchain-based applications has led to a rich set of architectural and software patterns that offer reusable solutions to recurrent design challenges (Six, Herbaut, and Salinesi, 2022a). These patterns help developers address soft goals–including security, scalability, transparency, and performance—while mitigating inherent limitations of blockchain systems, such as latency, scalability bottlenecks, and governance constraints.

Selecting appropriate patterns in blockchain-based applications remains a challenging and often ad hoc process. The growing number of available patterns, each exerting multiple and sometimes conflicting influences on soft goals (e.g., performance, security, or cost), complicates systematic decision-making. Unlike functional requirements, which can be verified against clear achievement criteria, soft goals are inherently subjective, relative, and trade-off–driven. This makes their evaluation and prioritization particularly difficult.

Currently, pattern selection often relies on the architect's experience rather than a structured methodology. This can lead to misalignment between business strategies and technical decisions, resulting in suboptimal system designs and unforeseen negative issues such as increased cost or reduced performance. These challenges highlight the need for a more rigorous, goal-oriented approach to pattern selection.

To bridge this gap, we propose **BC-TEAEM–Blockchain–Technology-Aware Enterprise Modeling**, a decision support framework that combines:

- an ontology modeling the relationships between blockchain patterns and trust-related soft goals.
- A knowledge-based reasoning engine capturing positive and negative influences of patterns on soft goals.
- a multi-criteria decision-making mechanism, implemented using TOPSIS ensuring recommendations align with user preferences.

The framework focuses on the interplay between a domain expert and a technical expert to ensure alignment and traceability. By iteratively capturing and refining preferences, BC-TEAEM supports systematic selection of blockchain patterns. We develop a prototype decision support tool implementing our method and validate it through a case study of a pharmaceutical company's supply chain traceability system, demonstrating the framework's applicability.

BC-TEAEM examines how business-level soft goals can be systematically connected to blockchain design patterns, with the aim of enhancing the transparency, traceability, and justifiability of architectural decisions. In this context, the following research questions (RQ_s) guide our investigation:

- *RQ_BC*$_1$*:* How can business-level soft goals be explicitly aligned with blockchain design patterns at the technical level?
- *RQ_BC*$_2$*:* How can the effects of specific technical decisions such as the selection of a blockchain pattern—be evaluated and systematically traced back to the originating business goals?
- *RQ_BC*$_3$*:* How can domain experts and technical experts effectively collaborate to ensure that solution design remains simultaneously business goal–driven and technology-aware?

The construction of the BC-TEAEM Knowledge followed a design science research methodology. The objective was to build a reusable artifact that is both theoretically grounded and practically useful for guiding trust-aware blockchain configurations. By following Wieringa's perspective on DSR (Wieringa, 2014), we ensure both practical relevance and scientific rigor. Our approach follows stages of the DSR cycle, from problem identification to Evaluation.

7.2 Problem Investigation & Research Gap

Unlike functional requirements, soft goals are subjective, context-dependent, and satisfied to varying degrees (Hu et al., 2015). The process of selecting patterns based on soft goals is further complicated by the fact that a single pattern can impact multiple soft goals differently. A pattern may positively impact some soft goals while negatively impacting others. For example, the On-Chain Encryption pattern improves security (by increasing confidentiality) but degrades performance (due to increased latency). The Off-Chain Data Storage pattern improves performance by reducing the amount of data that needs to be stored and processed on the blockchain, but decreases confidentiality because there is no security guarantee for external systems. This results in a trade-off.

A number of works have documented blockchain patterns. Authors in (Xu et al., 2018; Xu et al., 2025) present extensive catalogs organized into categories such as interaction, data management, security, and structural patterns. (Six, Correa-Restrepo, et al., 2022) extend this work by proposing a blockchain pattern ontology that formalizes structural relationships between patterns. (Besancon et al., 2022) propose a blockchain ontology for decentralized application (DApps) development, focusing on functional components rather than on non-functional trade-offs. While these resources are valuable, they primarily serve as descriptive references and do not connect patterns to soft goals in a way that supports automated reasoning.

Addressing this gap requires bridging structural representations with decision-making capabilities. From a requirements engineering perspective, soft goal modeling has been widely studied (Hu et al., 2015; Sumesh and Krishna, 2022), often using frameworks such as the NFR (Non-Functional Requirements) Framework to capture qualitative attributes and interdependencies. However, few works explicitly integrate these models into blockchain-specific architectural decision-making. The importance of managing trade-offs in blockchain ecosystems is highlighted by (Jovanovic et al., 2022), but without providing a formalized link between soft goals and patterns.

Decision-support tools for blockchain-related design have also emerged. Multi-Criteria Decision-Making (MCDM) techniques are applied to platform or configuration selection (Bahar et al., 2023; Soundararajan and Shenbagaraman, 2023), while TOPSIS is explored for blockchain sharding strategies (Liu et al., 2022). These approaches demonstrate the utility of quantitative trade-off analysis but typically rely on static attributes, lack integration with ontologies, and do not model inter-pattern or inter-soft goal relationships. Overall, existing approaches tend to address either:

1. **Pattern representation** — focusing on cataloging or structural ontologies without modeling impacts on soft goals,
2. **Soft goal modeling** — emphasizing qualitative attributes and interdependencies without linking to specific patterns.
3. **Decision-making** — applying MCDM methods without leveraging an ontology to capture complex interdependencies.

7.3 Artifact Design Methodology: Knowledge Base Construction

The first step was to gather information about existing blockchain patterns and the trust-related soft goals that they influence. At this stage the key decision was whether to rely on academic sources or to also include practical knowledge from industry. We opted for a combination pf both because academic literature provides conceptual clarity and systematic classifications, while industrial case studies reveal how patterns behave in real-world settings trade-offs are critical.

We use scientific literature as our primary source, relying on taxonomies, surveys, and studies on blockchain architectures, design patterns, and enterprise modeling. These works provide formal definitions, structured classifications, and comparative frameworks that offer a rigorous and coherent foundation for constructing ontology.

7.3.1 Selection of Blockchain Patterns

To identify relevant patterns, we began with catalogs proposed by (Xu et al., 2018; Xu et al., 2025) which provides a list of 15 blockchain patterns divided into four categories. These categories are widely recognized in the literature and provide a systematic starting point for modeling the design space::

- ***Interaction between blockchain and external world:*** patterns that describe how blockchain communicate with external systems. Examples include *Oracle*, *Reverse Oracle*, and the integration of *Legal Contracts* with *Smart Contracts*.
- ***Data Management Patterns:*** patterns that deal with the storage and organization of data, either on-chain or off-chain. Examples are *Encrypting On-Chain Data*, *Tokenization*, *Off-Chain Data Storage*, and *State Channel*.
- ***Security Patterns:*** patterns that enhance the protection of blockchain-based applications. Examples include *Multiple Authorization*, *Off-Chain Secret Enabled Dynamic Authorization*, and *X-Confirmation*.
- ***Structural Patterns of Contract:*** patterns that define dependencies and behaviors between smart contracts, such as *Contract Registry*, *Data Contract*, *Embedded Permission*, *Factory Contract*, and *Incentive Execution*.

While this catalog provides a strong foundation, it does not fully capture the breadth pf blockchain design patterns options available today. To address this gap, We extended this set with variants of patterns reported in more recent studies (Rajasekar et al., 2020; Henry and Tucci-Piergiovanni, 2024) and industrial practice. For example, patterns such as *Centralized Oracle* or privacy-preserving mechanisms based on *Zero-Knowledge Proofs* have gained practical relevance but were not covered in earlier catalogs.

A representative subset of these patterns is shown in Table 7.2, while the complete list and detailed descriptions can be found in (Xu et al., 2018). This extended collection forms the basis for the BC-TEAEM ontology, ensuring that our knowledge base reflects both established research and current industrial practice.

7.3.2 Selection of Soft Goals

In requirements engineering, a distinction is often made between *hard goals* and *soft goals*. Hard goals represent functional requirements that a system must satisfy in clear and verifiable manner (e.g., "record every shipment in the blockchain"). Soft goals, by contrast, represent non-functional requirements or qualitative properties that influence a system's quality and trustworthiness but cannot be fulfilled in binary way (Hu et al., 2015). Statements such as "the system should be secure" or "transactions should be fast" guide design decisions even though their satisfaction is inherently gradual rather than absolute.

Soft goals function as quality criteria that systems and their components are expected to meet. They are subjective (context-dependent), relative (improving one may degrade another), diverse (systems emphasize different qualities), and interdependent (trade-offs are common). These characteristics make soft goals essential yet challenging to model, particularly in blockchain-based applications where design tensions are pervasive (Hu et al., 2015).

We provide a contextualized definition of soft goals tailored to blockchain applications, and more concretely to our running example of drug traceability in the pharmaceutical supply chain. The soft goals considered in our framework draw on established requirements engineering approaches such as the NFR Framework, the ISO/IEC 25010 quality model, as well as blockchain-specific literature (Kambilo et al., 2023). From these sources, we selected soft goals that are both frequently cited in research and commonly encountered in practice when designing blockchain systems:

- **Cost:** referring to both transaction fees and infrastructure expenses, including deployment and ongoing maintenance. Cost is a critical determinant of adoption in enterprise contexts.
- **Integrity:** ensuring data consistency, immutability, and verifiability across the distributed ledger. Integrity is at the heart of blockchain's value proposition.
- **Performance:** the ability to process transactions efficiently at scale. Performance includes transaction throughput, latency, and the system's ability to handle network congestion.
- **Interoperability:** the capacity of the blockchain system to interact seamlessly with external systems such as IoT devices, traditional databases, or other blockchains. This is especially relevant in supply chain ecosystems that involve multiple heterogeneous actors.
- **Privacy:** the ability to restrict access to sensitive data while allowing transparency where appropriate. Privacy mechanisms must balance confidentiality with regulatory requirements for traceability.
- **Security:** encompassing protection against a broad spectrum of threats, including cyberattacks, cryptographic vulnerabilities, and unauthorized access. Security ensures trust in the reliability of the blockchain network.

- **Transparency:** providing visibility and auditability of transactions for all authorized stakeholders. Transparency is a cornerstone of trust in inter-organizational processes.

Taken together, these soft goals represent the dimensions of trust that guide the selection and evaluation of blockchain patterns in our framework. They serve as the criteria against which different design alternatives can be compared and balanced, ensuring that final configurations align with both technical feasibility and business priorities.

7.3.3 Modeling the Relationships between Soft goals and Patterns

Several approaches could have been considered, such as taxonomies, knowledge graphs, or ontologies to model this relationship. We choose to build an ontology for two main reasons: First, an ontology allows us to explicitly represent not only concepts (patterns and soft goals) but also the relationships between them (positive, negative, or neutral influence). Second, ontologies can be queried and reasoned upon by software agents, making them more powerful than a simple taxonomy.

To capture the relationship between blockchain patterns and soft goals, we conducted an engineering analysis grounded in the foundational works of (Xu et al., 2018; Xu et al., 2025). Their studies provide valuable contextual insights and taxonomic classifications, but they do not include formal representation suitable for automated reasoning.

The key challenge lies in representing how a given pattern influences one or more soft goals, and in doing so, capturing both its benefits and potential drawbacks. To achieve this, we adopted a scoring process based on a five-point ordinal scale (Table 7.1). Each pattern–soft goal relationship is assigned a score reflecting whether the influence is strongly positive, moderately positive, neutral, moderately negative, or strongly negative. This allows us to quantify otherwise qualitative insights in a manner suitable for systematic comparison and decision support.

The scoring process follows a structured sequence:

1. Review literature descriptions (e.g., Xu et al., 2018; Xu et al., 2025) of each pattern's motivating forces and consequences.
2. Identify both direct and indirect effects on each selected soft goal.
3. Assign influence scores according to the criteria in Table 7.1.
4. Where the evidence is ambiguous, validate the assessment through expert consultation or workshop discussions.

Symbol	Influence Type	Definition
++	Strong Positive	The pattern significantly improves the soft goal under typical usage scenarios.
+	Moderate Positive	The pattern improves the soft goal, but the gains are limited or highly context-dependent.
0	Neutral / None	The pattern has negligible or no measurable impact on the soft goal.
-	Moderate Negative	The pattern somewhat hinders the soft goal, with limited severity.
–	Strong Negative	The pattern significantly worsens the soft goal.

TABLE 7.1: Scoring Process: Influence of Patterns on Soft Goals

To ensure clarity and reproducibility, each influence category is grounded in explicit semantic rules:

- **Improves:** A pattern positively contributes to achieving a soft goal by enhancing its properties or facilitating its implementation. (For i.e: The Oracle Pattern improves Interoperability by allowing blockchain systems to communicate with external systems).
- **Worsens:** A pattern negatively impacts a soft goal, making it harder to achieve or introducing trade-offs.
- **Neutral:** A pattern has no significant impact on a soft goal, meaning it neither improves nor worsens it.

These soft goal are encoded by using the scoring scheme defined in Table 7.2, enabling systematic comparison across patterns. The outcome of this procedure is a structured knowledge base describing how each blockchain pattern influences each soft goal.

Unlike descriptive catalogs, this representation is engineered for computational reasoning: it can be directly integrated with multi-criteria decision-making techniques—particularly TOPSIS—to quantify trade-offs and guide architectural choices.

Table 7.2 summarizes the encoded relationships between selected patterns and soft goals. For example, Oracle patterns strongly improve interoperability but moderately worsen security due to external data dependencies. In contrast, Encrypting On-Chain Data strongly improves privacy but moderately worsens performance. Notably, some patterns (e.g., Factory Contract) have largely neutral effects on most soft goals, suggesting they may be selected with minimal trade-offs. These observations highlight both expected and counter-intuitive relationships; e.g., Off-Chain Data Storage improves performance but can weaken integrity, a trade-off often overlooked in practice. Such

Patterns	Soft Goals						
	Cost	**Interop.**	**Privacy**	**Security**	**Integrity**	**Trans.**	**Perf.**
Oracle	N	I	W	W	N	W	W
Reverse Oracle	N	I	N	W	N	N	I
Legal and Smart Contract	N	N	N	N	I	I	N
Encrypting On-Chain	N	N	I	W	N	N	W
Tokenization	I	N	N	W	I	I	N
Off-chain Data Storage	I	N	N	W	I	W	I
State channel	I	N	I	W	N	I	I
Multiple Authorization	W	W	N	I	I	N	I
Off-chain Secret	W	I	N	W	N	N	I
X-Confirmation	N	N	N	I	I	I	N
Contract Registry	W	I	N	I	I	N	I
Data Contract	I	N	N	I	N	N	I
Embedded Permission	N	N	N	I	N	I	W
Factory Contract	W	N	N	I	N	N	I
Incentive Execution	I	N	N	N	N	N	N

TABLE 7.2: Relationships Between Patterns and soft goals

Legend: I = Strong and Moderately Improves; W = Strong and Moderately Worsens; N = Neutral. The abbreviations: Interop. (Interoperability), Trans. (Transparency), Perf. (Performance).

insights motivate the need for systematic reasoning in pattern selection.

7.3.4 Modeling Trade-offs Between Patterns and Soft Goals

Patterns may be mutually supportive or conflicting, making it essential to capture both inter-pattern and inter–soft goal relationships. Building on (Six, Herbaut, and Salinesi, 2022a), we extend existing relationship types with:

- **Benefits from:** one pattern may reinforce or complement another. For example, the Decentralized Oracle pattern can benefit the Off-Chain Data Storage pattern by providing a secure and reliable way to fetch external data, thereby enhancing the system's interoperability and performance.
- **Conflicts with**: certain patterns may constrain the use of others. Their combination can lead to contradictory systems or design inconsistencies. This type of relationship is useful for identifying potential design trade-offs, which are important in pattern recommendation. For example, Encrypting On-Chain Data improves privacy, while Reverse Oracle reduces it by exposing data externally.

7.3.5 MCDM Selection

To support trade-off resolution and pattern ranking, we opt for the Technique for Order of Preference by Similarity to Ideal Solution (TOPSIS) (Papathanasiou et al., 2018) as the Multi-Criteria Decision-Making method due to its simplicity, robustness, and relevance in goal-oriented trade-off analysis. TOPSIS is a well-established multi-criteria decision-making (MCDM) method designed to rank alternatives that exhibit attributes of different scales and types (Six, Herbaut, and Salinesi, 2022a).

From a decision matrix and a set of weights representing user preferences, TOPSIS computes for each alternative a score ranging between 0 and 1. This score reflects the relative closeness of the alternative to the ideal solution, which is composed of the best possible values for each criterion. Conversely, the algorithm also considers the negative ideal solution, which aggregates the least desirable values. The outcome is a ranking of alternatives based on their distance from these two poles.

Step 1: Construction of the Decision Matrix: The decision process begins with the formulation of the decision matrix:

$$D = \begin{bmatrix} x_{11} & x_{12} & \dots & x_{1n} \\ x_{21} & x_{22} & \dots & x_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ x_{m1} & x_{m2} & \dots & x_{mn} \end{bmatrix} \tag{7.1}$$

Where:

- m denotes the number of alternatives (blockchain patterns);
- n denotes the number of criteria (soft goals);
- x_{ij} represents the performance of alternative i with respect to softgoal j.

Step 2: Normalization and Weight Application: Since the criteria may be expressed on different scales, the matrix is normalized and weighted according to user preferences:

$$v_{ij} = w_j \times r_{ij} = w_j \times \frac{x_{ij}}{\sqrt{\sum_{i=1}^{m} x_{ij}^2}} \tag{7.2}$$

Here, w_j corresponds to the weight assigned to soft goal j, reflecting the user's subjective prioritization and perception of trust in how patterns influence soft goals.

Step 3: Determination of Ideal and Negative-Ideal Solutions: For each soft goal, the algorithm identifies the best and worst performances across alternatives:

$$v_j^* = \max(v_{ij}), \quad v_j^- = \min(v_{ij}) \tag{7.3}$$

The distance of each alternative from these two reference points is then calculated as:

$$S_i^* = \sqrt{\sum_{j=1}^{n} \left(v_{ij} - v_j^*\right)^2}, \qquad S_i^- = \sqrt{\sum_{j=1}^{n} \left(v_{ij} - v_j^-\right)^2} \tag{7.4}$$

Step 4: Calculation of the Relative Closeness: Finally, the relative closeness of each alternative to the ideal solution is computed as:

$$C_i^* = \frac{S_i^-}{S_i^* + S_i^-} \tag{7.5}$$

A higher C_i^* indicates that the corresponding pattern better satisfies the weighted set of soft goals.

Interpretation: The resulting scores provide a ranking of blockchain patterns aligned with user-defined priorities. For instance, if scalability and security are weighted more heavily, the algorithm may recommend combining *Off-Chain Data Storage* with *On-Chain Encryption*, while making explicit potential trade-offs such as reduced performance. In this way, TOPSIS operationalizes the trade-off analysis between soft goals and provides architects with a systematic decision-support tool for selecting blockchain patterns.

Unlike qualitative ranking methods, TOPSIS quantitatively evaluates each pattern by calculating its geometric distance from both an ideal and an anti-ideal solution, based on user-defined soft goal weights. This makes it particularly suitable for scenarios involving conflicting non-functional requirements.

TOPSIS is especially adapted in our context for the following reasons: (a) It supports dynamic user input and real-time re-ranking of patterns as stakeholder preferences evolve. (b) It aligns with the concept of soft goal trade-offs by capturing closeness to an ideal solution. (c) It also produces a continuous score that facilitates the ranking and visualization of multiple pattern alternatives.

7.4 Artifact Conception: Design of the Knowledge Base

BC-TEAEM consists of two interrelated artifacts: (1) a domain-specific ontology modeling the relationships between blockchain patterns and soft goals, and (2) an interactive tool that operationalizes this ontology for pattern recommendation.

7.4.1 Conceptual Ontology

The ontology is designed using OntoUML (G. Guizzardi et al., 2015), a well-founded language grounded in the Unified Foundational Ontology (UFO). ontoUML provides ontological distinctions such as kinds, roles, relators, and events, enabling precise modeling of domain concepts and their interdependencies.

We extend the blockchain pattern ontology of (Six, Correa-Restrepo, et al., 2022), which models structural relationships (e.g., requires, variant of, related to), by incorporating soft goals and explicit influence relations. This extension allows the ontology to support reasoning about trust-related qualities and architectural trade-offs.
We extend it by:

- Adding impact relationships between patterns (or their variants) and soft goals, with direction and magnitude encoded using the scoring criteria.
- Introducing inter-pattern relationships: "Conflict With" and "Benefits From".

We select OntoUML because it provides a rigorous ontological foundation grounded in the Unified Foundational Ontology (UFO) standard, enabling precise modelling of categories, roles, and relational structures. This capability is essential for representing patterns, soft goals, and their interdependencies in a semantically coherent manner. In contrast, other ontology engineering method such as NeOn (Suárez-Figueroa et al., 2011), while valuable for supporting Collaborative development and the reuse of ontology networks, adopt a more process-oriented focus. For the purposes of this research, OntoUML's capacity to capture the semantics and constraints of trust-related concepts makes it a more suitable choice.

We applied OntoUML to formalize:

- *Patterns:* a kind representing blockchain design patterns from the literature.
- *Variants:* subtypes of patterns capturing design alternatives (e.g., centralized vs. decentralized oracle).
- *Soft goal:* qualities such as Privacy, Transparency, Performance, Security, or Interoperability.

- *Impact (Relator):* mediates the relationship between a pattern (or variant) and a soft goal, annotated with an influence polarity (improves, worsens, neutral) and quantified intensity.
- *Trade-off Relations:* inter-pattern links (Benefits_From, Conflicts_With) and inter–soft goal dependencies (enhances, reduces).

This ontological grounding ensures that the resulting model is not only structurally consistent but also semantically rigorous, supporting later reasoning and decision-making tasks such as pattern recommendation. The different class of our ontology is explained in the next sub-section.

7.4.2 Ontology Overview

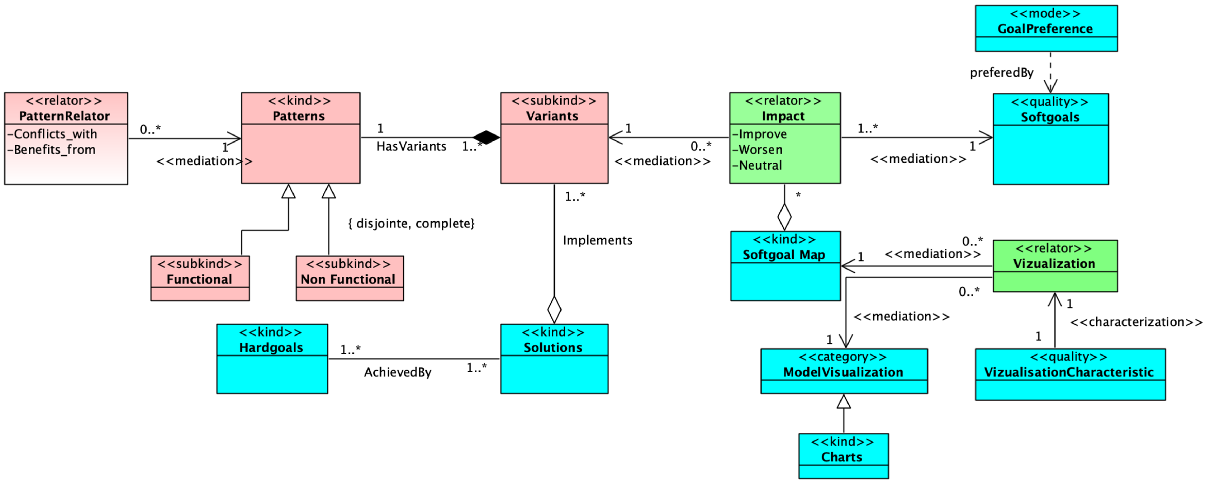

FIGURE 7.1: This figure illustrates the ontology of the relationship between the blockchain patterns and the soft goals.

To answer to our RQ_BC$_1$, the first outcome is the creation of TEAEM ontology, which formalizes the relationships between soft goals (business goals) and patterns, while also addressing the trade-offs among patterns (Top-Down Alignment). As shown in Figure 7.2, the driving idea behind the ontology is:

- The base structure (highlighted in pink) is derived from the ontology proposed by Six et al. (Six, Correa-Restrepo, et al., 2022). It includes foundational concepts such as Pattern, variant, and their general relationships.
- Our Contribution extends this foundation through additional components (blue and green) that model trade-offs between patterns and integrate soft goals and their influence. At core of the ontology is the `Impact` Relator, which links pattern variants to soft goals, and their types of influence.

To illustrate the different classes of our ontology and how this ontology supports decision-making, we refer to our running example introduced in sub-section 7.2.

Patterns Class: represents blockchain patterns (Table 7.2), organized into two categories:

- *Functional patterns:* Core capabilities (for example, Oracle patterns for external data).
- *Non-functional patterns:* Quality attributes (for example, Multiple Authorization for security).

A scenario in which architects employ the Oracle pattern to integrate off-chain data illustrates how variant selection can be guided by the ontology. Two variants are typically considered:

- Centralized oracle: Improves performance and reduces cost, but reduces security due to a single point of failure.
- Decentralized oracle: Improves security but worsens performance due to additional validation steps.

The ontology can detect such conflicts automatically. By using inference, it is possible to identify conflicts directly between variants of patterns based on their opposing effects on soft goals, thus generating new knowledge about pattern conflicts.

$$\begin{aligned} \forall(Oracle) \in P,\ (v_1, v_2) \in V,\ security \in S : \\ \text{Improves}(v_1, security) \wedge \text{Worsens}(v_2, security) \wedge \\ \text{hasVariant}(Oracle, v_1) \wedge \text{hasVariant}(Oracle, v_2) \\ \Rightarrow \text{conflict}(v_1, v_2, s) \end{aligned}$$

This rule captures a scenario where two variants of one pattern Oracle, v_1 and v_2, exert opposing influences on a given soft goal s. Variants v_1 improves s, while v_2 worsens it. If the condition holds, a conflict exists between variants of patterns v_1 and v_2 regarding the soft goal s.

Soft Goals Class: represents qualitative attributes such as performance and security. Each ***Variant*** of pattern is connected to ***Soft goals*** through the ***Impact*** Relator. These links are interpreted in light of user preferences.

A situation in which an enterprise assigns higher priority to *Security* than to *Performance*. A decentralized oracle improves security while simultaneously worsens performance. According to the ontology's reasoning process, the trade-offs introduced by a decentralized oracle variant are considered acceptable. This implies that the benefits of enhanced security outweigh the drawbacks of reduced performance. Therefore, this variant is recommended.

$$\begin{aligned}
&\forall v \in V, s_1, s_2 \in S : \\
&(\text{Improves}(v, s_1) \vee \text{Improves}(v, s_2)) \vee \\
&(\text{Improves}(v, s_1) \wedge \text{Worsens}(v, s_2) \wedge \text{Pref}(s_1, s_2)) \Rightarrow \text{Recommended}(v)
\end{aligned}$$

This demonstrates how the ontology enables preference-aware trade-off analysis in pattern selection.

For Reasoning on Pattern Interactions, in addition to pattern-soft goal relationships, the ontology models inter-pattern relationships. Some patterns have synergistic impacts; others are conflicting. For example, Encrypting On-chain Data improves privacy but degrades performance, and Off-chain Data storage improves performance but may weaken data transparency. Their combined selection creates a conflict effect, which the ontology can detect automatically. The reasoning process helps blockchain architects spot potential conflicts early to prevent unexpected issues in the final design of the system.

The outcome of the reasoning process is a set of recommended solutions. These are captured in the ***Solutions*** class, which encapsulates the concrete implementation of patterns and their variants. All impacts are aggregated and structured within the ***Softgoal Map*** class, instantiated in the ontology, and linked to higher-level goals represented by the ***Hardgoals*** class.

7.4.3 Ontology Implementation

After constructing the ontology conceptually, the next step was its technical implementation. We use **Protégé**, an open-source platform that supports the development of domain models and knowledge-based applications. The implementation serves two main purposes: first, to make the ontology computable by encoding it in the formal OWL 2 representation language; and second, to prepare it for reasoning tasks, including consistency checking, querying, and automated classification.

Translation from OntoUML to OWL: OntoUML provides the conceptual foundations, but implementation requires translating its constructs into OWL. Following the guidelines of (G. Guizzardi et al., 2015), we map Kinds and Sub-Kinds of patterns to OWL classes (such as *Pattern*, *Oracle*, or *Encrypting On-Chain*), and we represent soft goals as OWL classes corresponding to trust-related qualities such as (*Security*, *Transparency*,

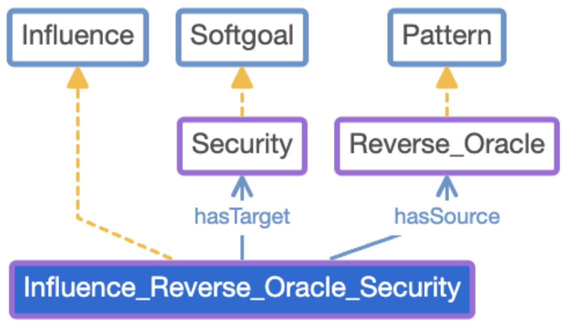

FIGURE 7.2: This figure illustrates the view implementation of ontology in Protégé

or *Privacy*). Contribution *Relators* are translated into object properties linking patterns and soft goals, with annotations capturing the polarity (positive/negative/neutral) and intensity. Dependencies among soft goals are also modeled as object properties with restrictions (e.g., reduces, enhances).

Implementation in Protégé: The ontology was encoded using OWL 2.0 using Protégé 5.6, which provides a mature environment for editing ontologies and performing reasoning tasks. The implementation involved several steps:

- **Model the taxonomy of blockchain patterns:** classes were created for Pattern and its variants (Oracle, Centralized Oracle).
- **Define Soft goals** as classes representing qualities and dispositions, such as Performance, Transparency, and Security.
- **Create object properties:** Such as Benefits from, Improves, Worsens to represent influence relations between patterns and soft goals. Contribution polarity and intensity were added through annotation properties.
- **Reasoning:** The ontology was checked with reasoners (HermiT and Pellet) to ensure logical consistency. Automated classification verified that pattern subkinds were correctly inferred under their parent kinds, and that dependency relations among soft goals respected transitivity and disjointness constraints.

Implementing the ontology in Protégé adds machine-interpretable semantics to the conceptual model. It allows Programmatically exploitable, enabling integration with the BC-TEAEM framework. This allows us to query the ontology using SPARQL, for example, "all patterns that negatively affect Performance but positively contribute to Security." It helps also for supporting expert reviews and ontology engineering best practice.

7.4.4 Ontology Validation

The Ontology Validation ensures that the model is internally sound, relevant to the domain, and capable of supporting meaningful reasoning tasks. Validation is conducted along three axes: (i) *ontology validity*, assessing internal correctness and conformance to OntoUML/UFO; (ii) *ontology relevance*, verifying that the model captures the required concepts for blockchain pattern analysis; and (iii) *results and analysis*, demonstrating that the ontology supports useful reasoning over instantiated data.

Ontology Validity: We ensure the ontology validity via a three-fold process:

- careful alignment with OntoUML/UFO constraints,
- systematic conceptual review of class hierarchies and mediated relations,
- automated consistency checking using OWL reasoners (HermiT and Pellet).

This guarantees the correct use of core constructs—including kinds, sub-kinds, relators, mediation constraints, disjointness, and generalization rules.

Ontology Relevancy: Relevance was assessed using a set of competency questions (CQs) designed to test whether the ontology supports trust-aware pattern selection tasks. Examples include:

- CQ_1: Which patterns improve Security?
- CQ_2: Which patterns introduce performance penalties?
- CQ_3: Which pattern combinations generate conflicts?
- CQ_4: How do variations in privacy preferences affect recommended patterns?

These questions were evaluated by SPARQL queries. For example:

LISTING 7.1: Retrieving patterns that improve Security

```
PREFIX teaemontology:
   <http://www.semanticweb.org/eddy/teaemontology#>

SELECT DISTINCT ?pattern
WHERE {
  ?rel a teaemontology:Influence ;
       teaemontology:hasSource ?pattern ;
```

```
        teaemontology:hasTarget teaemontology:Security ;
        teaemontology:hasInfluenceType teaemontology:Improves .

  ?pattern a teaemontology:Pattern .
}
ORDER BY ?pattern
```

Results and Analysis: To evaluate practical usefulness, the ontology was instantiated with representative patterns (Centralized Oracle, Decentralized Oracle, Push-Based Oracle) and soft goals (Performance, Privacy, Transparency, Security). Each Impact instance recorded.

- polarity (improves / worsens / neutral), and
- scalar intensity values in the [-2, +2] Likert-like scale used by the scoring model.

These instances support the BC-TEAEM evaluation matrix that is later processed by the TOPSIS module. Reasoning tests confirmed that:

- conflicts between oracle variants were correctly inferred,
- soft-goal dependencies were propagated appropriately,
- contribution structures aligned with expert expectations.

This validates the ontology's ability to support explainable and trustworthy recommendations.

7.5 Artifact Development: BC-TEAEM Recommendation Engine

7.5.1 Architectural Overview

The BC-TEAEM Recommendation Engine is structured as a layered architecture that transforms user preferences into well-justified blockchain pattern recommendations. Building on the knowledge base, it delivers an operational decision-support mechanism that guides stakeholders through a structured reasoning process accounting for preferences, dependencies, and trade-offs to operationalize trust.

At the high level, the architecture is composed of five interacting components

- ***User Preference Input Layer***: Users specify the soft goals most relevant to their context (e.g., prioritizing Security over Performance).
- ***Soft Goal Influence Propagation Engine***: User preferences are automatically adjusted according to inter-soft-goal dependencies encoded in the ontology. For

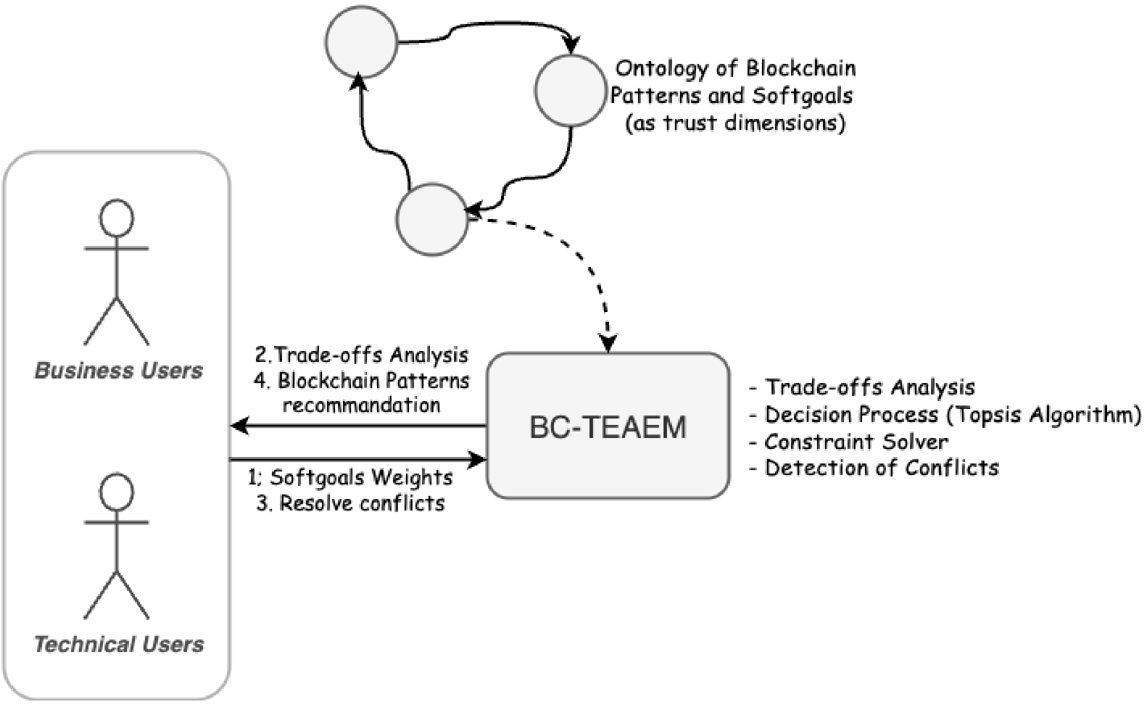

FIGURE 7.3: Overview of TEAEM Recommendation Engine

instance, increasing Transparency may reduce Performance, while reinforcing Security may indirectly enhance Scalability.

- ***Evaluation Matrix Constructor***: This module retrieves pattern–soft goal relations from the ontology and builds a matrix where patterns are rows and soft goals are columns. Qualitative effects (Improves / Worsens / Neutral) are converted into numerical scores and weighted by adjusted preferences, forming the input for decision analysis.
- ***Decision-Making Module***: Using TOPSIS, the system normalizes the evaluation matrix, identifies ideal best and ideal worst solutions, and ranks patterns accordingly. For example, a pattern that improves high-priority goals and only slightly worsens low-priority ones will emerge near the ideal solution.
- ***Visualization and Feedback Layer***: Results are presented through interpretable visualizations, such as ranked lists, color-coded impacts, and conflict indicators. If two recommended patterns are mutually exclusive, the interface explicitly highlights this contradiction to guide informed choices.

Taken together, these five components provide a coherent architecture that bridges the conceptual ontology of patterns and soft goals with the practical need for decision-support. The following section details the decision support model (methodology) that governs each com

7.5.2 Operational Workflow: Decision Support Model

This subsection presents the decision process embedded in the BC-TEAEM tools to identify the most appropriate blockchain patterns according to user-defined soft goals in order to increase (improve) trust. Figure 7.4 provides an overview of the process. First the practitioner specify the input requirements in form of soft goals preferences 7.5.2. The recommendation engine also relies on knowledge base (ontology) that encodes blockchain design patterns with their relationships with soft goals. Based on these two inputs, the decision process **??** applies the TOPSIS multi-criteria algorithm to compute a ranking of patterns. The resulting recommendation highlights the patterns that best align with the practitioner's priorities while making explicit the trade-offs among conflicting soft goals.

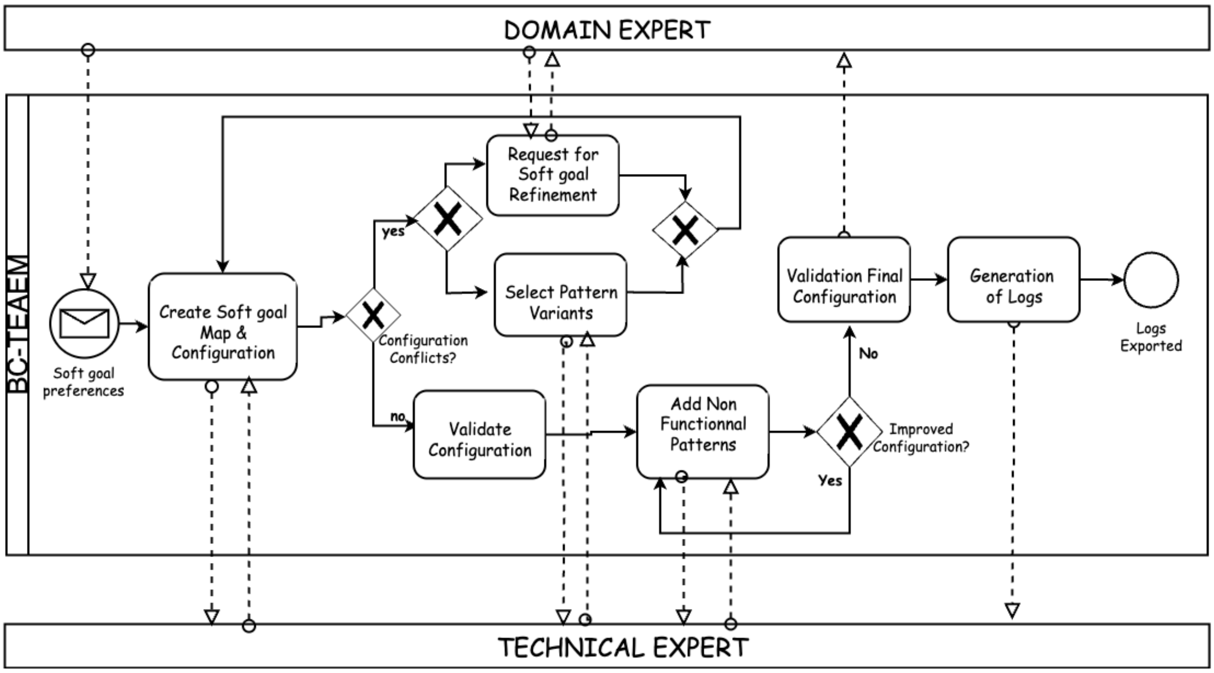

FIGURE 7.4: This figure illustrates the workflow of the applications by considering users' roles and tools features.

Capturing Softgoals Preferences:

The recommendation process begins with the collection and structuring of input data. They key inputs are the soft goals (performance, security, scalability, or cost) and their relative priorities as provided by the stakeholders. These express the high-level expectations that shall all subsequent evaluations.

We distinguish two types of users:

- *Domain Experts (DE)*: define soft goals, priorities, and trade-offs.

- *Technical Experts (TE)*: refine, validate, and implement feasible pattern-based solutions based on DE input.

The interaction between Domain Experts and Technical Experts forms the foundation of the decision process. DE provides the strategic vision that the enterprise want to achieve, while TE ensure that this vision remains technically achievable. In the prototype, these inputs are captured through the Explorer Module, which enables users to inspect patterns, their relationships to soft goals, and assign priorities.

- *Improves:* means that the domain expert is interested in patterns that positively contribute to a specific soft goal. (weights=$+2$ (for strongly agree) and $+1$ (for agree))
- *Best effort:* Attempt to improve a soft goal; if not feasible, consider alternatives (weight adjusted automatically from $+2$ or $+1$ to 0, -1, or -2.)

This step is the most important because soft goals often compete with one another. For example, transparency may be a dominant concern, but stakeholders may still wish to balance it against cost efficiency. By capturing user preferences, the system ensures that recommendations remain tailored to the decision-maker's context rather than relying on generic or predefined assumptions.

Influence Propagation:

Soft goals often compete, and their prioritization must reflect the decision context rather than generic assumptions. As discussed in the previous chapter, the relationships between soft goals and patterns are essential because they determine pattern suitability; some patterns are complementary, while others are incompatible. The *softgoal influence propagation engine* applies the dependency rules encoded in the ontology to adjust initial weights, increasing or decreasing the importance of related soft goals according to positive, neutral, or negative influences.

This module generates a visual **soft goal Map (SMp)** that illustrates how patterns align with prioritized soft goals. Based on this map, the engine supports three recommendation cases:

- *Single Match:* One single solution in the SMp, the tool recommends it.
- *Multiple Matches:* If multiple patterns are identified, the TE refines the recommendation using domain knowledge.
- *Conflicting Goals:* If trade-offs exist (example, security vs performance), the DE is consulted to prioritize soft goals, and the recommendation is adjusted accordingly.

7.5.3 Resolving Trade-offs and Refining Solutions

To address trade-offs among competing soft goals, our approach relies on an iterative refinement. When the relative importance of soft goals (e.g., performance vs security) can not be determined by the TE, the DE IS consulted to provide guidance (prioritization of soft goals). The tool then recalculates the soft goal satisfaction map and updates the recommended configuration. This stage typically involves DE adjusting priorities or requirements and TE assessing feasibility.

Generating Recommendations and Explanations: If no conflicts are detected, the system validates the configuration by constructing an evaluation matrix from the ontology, translating qualitative pattern impacts into quantitative values, and weighting them using propagated preferences. The initial configuration contains only the functional patterns required for the core application logic.

Technical experts may then add non-functional patterns to strengthen alignment with prioritized soft goals. These patterns are presented as visual cards summarizing their contributions. The engine checks whether additions improve the configuration and iterates until a satisfactory solution is reached. Patterns are then evaluated and ranked using the TOPSIS algorithm, which computes scores based on their contribution to selected goals.
Once a satisfactory configuration is obtained, the tool may suggest complementary patterns to make solution more trustworthiness. The final output includes ranked recommendations, visual explanations, and graphical trade-off representations. n parallel, the system produces logs that document the reasoning steps, ensuring traceability for audits or future refinements.

7.5.4 Implementation of BC-TEAEM

BC-TEAEM is implemented as a prototype integrating the ontology of patterns and soft goals, a decision-making module combining weight propagation and TOPSIS, and an interactive interface. The system serves both as a proof-of-concept and as an environment for exploring how preferences, dependencies, and trade-offs shape pattern recommendations. Its implementation relies on three components: ontology management, decision analysis computation, and user interaction.

Ontology Management

The ontology developed in the previous chapter forms the backbone of the system. Authored in Protégé and exported in OWL, it is queried programmatically through

Python. It provides:

- pattern–soft goal relations (improves, worsens, neutral),
- inter-pattern and inter–soft goal dependencies used for propagation,
- extensibility for adding new concepts without altering algorithms.

Decision Analysis Computation

The computation layer, implemented in Python 3.13, comprises:

- **Influence Propagation Module**: interprets ontology-based dependencies and adjusts user weights accordingly,
- **TOPSIS Module**: applies normalization, ideal-solution computation, and closeness ranking to generate pattern recommendations.

This modular architecture maintains separation of concerns and supports future extension.

User Interaction and Visualization

The interface, built with Streamlit, enables preference elicitation, real-time propagation feedback, and recommendation generation. Visual outputs include:

- preference settings converted into weights,
- soft goal maps reflecting propagated influences,
- ranked pattern recommendations,
- tables, Graphviz graphs, and narrative explanations.

These visualizations ensure transparency and avoid a "black-box" effect.

Extensibility and Reusability

The prototype is modular and can incorporate new domains, soft goals, or patterns. An online demo is available[1], and the source code is accessible on GitHub[2].

In sum, BC-TEAEM demonstrates the effective combination of ontological reasoning, weight propagation, and multi-criteria decision analysis in a user-oriented system. The next section presents its operation through a concrete use case.

[1] https://demo.ontoteaem.fr/
[2] https://github.com/CRI-Collab/ontoTEAEMS

7.6 Running Example

To demonstrate the operational value of the BC-TEAEM tool, we present an illustrative use case scoped in the domain of Supply Chain Management Systems (SCMS). This domain is particularly suitable because supply chains are inherently inter-organizational, involving manufacturers, logistics providers, retailers, and regulators. In such settings, Trust is an important and recurring challenge. Stakeholders must exchange information and coordinate operations without necessarily having complete visibility into each other's processes or full confidence in their integrity. Blockchain technology has emerged as a promising solution to enhance trustworthiness, but its adoption requires careful selection of design patterns to balance competing objectives.

7.6.1 Scenario Context - AVIA Logistics

A large logistics enterprise is tasked with managing the transportation of goods across multiple countries. The company is considering a blockchain-based solution to improve coordination with partners while ensuring regulatory compliance. Based on an internal stakeholder workshop, three soft goals were prioritized:

- **Performance Improvement:** ensuring that throughput and latency remain acceptable despite the high transaction volume in logistics operations.
- **Security Enhancement:** safeguarding the integrity and confidentiality of shipment data shared across the network.
- **Cost Reduction:** minimizing operational and infrastructure costs associated with blockchain adoption.

The stakeholders acknowledge that these goals may conflict: security mechanisms such as advanced encryption may increase costs, while performance optimization may sometimes compromise security guarantees.

7.6.2 Capturing Soft goals Preferences

During the initial phase, as specified in Figure 7.5 stakeholders indicate that Performance, Security and Cost Reduction in their priority (button improves) These preferences reflect the consortium's emphasis on building consumer trust through visibility while safeguarding sensitive information.

The interface captures these preferences through buttons, and the system normalizes the values to prepare them for analysis.

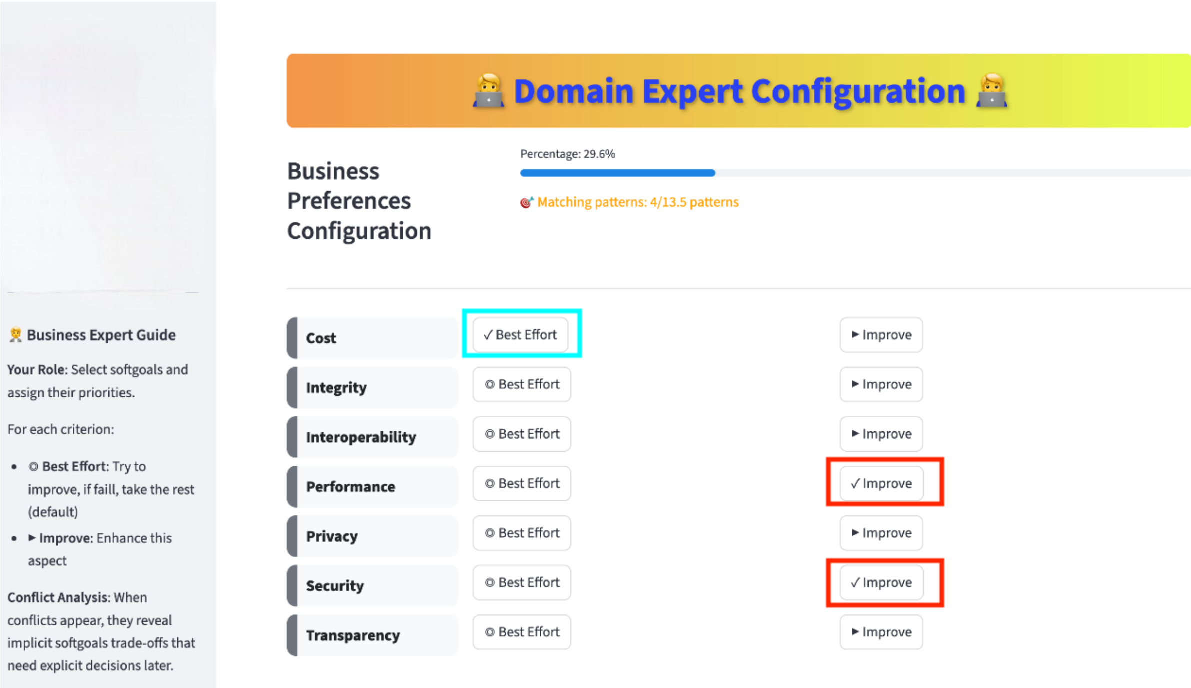

FIGURE 7.5: Capturing Soft goals Preferences on BC-TEAEM

7.6.3 Influence Propagation

The propagation engine adjusts the weights according to the interdependencies modeled in the ontology. For example, the ontology encodes a negative influence from enhanced security to cost, meaning that strengthening security typically increases expenses. As a result, the effective priority of cost reduction is slightly lowered. Conversely, the positive influence between performance and cost efficiency amplifies the importance of performance, since better throughput can reduce operational overhead.
In our specific use case, one of the recommended patterns is **s Off-Chain Data Storage**, together with its variant *Off-Chain Data Storage Cryptograhic Proof*. The tool also identifies other candidate patterns, but some of them conflict with the prioritized soft goals. For example: the *Centralized Oracle*, which improves performance and cost (2/3) but worsens security, and the *Decentralized Oracle*, which enhances security(1/3) but negatively impact both cost and performance. Such situations illustrate that is often impossible for business users to simultaneously satisfy all predefined soft goals. The recommendation engine therefore notices that to Business users and encourages them to prioritize among the soft goals, allowing the system to adapt its recommendations accordingly. Concretely, it is not feasible to select a single or set of pattern that simultaneously improves security, performance, and cost. Instead, the decision-making process requires stakeholders to establish a hierarchy of priorities so that the engine can identify the most suitable compromise in line with their strategic goals.

The Business Users are informed whenever conflicting recommendations arise, and they

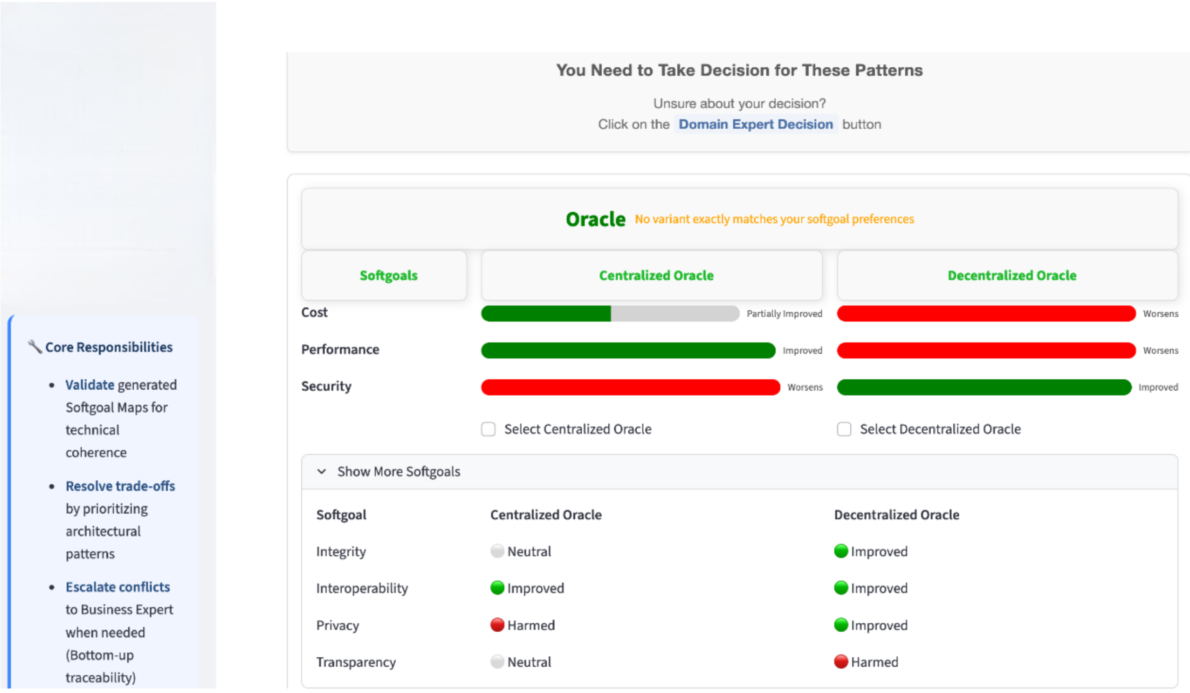

FIGURE 7.6: Generated Soft goals Maps on BC-TEAEM

must decide which soft goal to prioritize. To support this process, BC-TEAEM proposes a *Conflict Resolution Panel*, where alternative resolutions are presented through radio-button options. This interactive panel enables the stakeholders to explicitly select and validate their preferred compromise. Once a choice is made, the system incorporates it into the configuration, ensuring that the final reflects the prioritized soft goal and generates an adapted soft goal profile that guides the subsequent recommendation.

During this process, the first evaluation matrix is constructed in the background. This matrix serves as the analytic backbone of the recommendation engine. Each row of the matrix corresponds to a blockchain design patterns (or one of its variants), and each column corresponds to a soft goal. The entries are filled with the quantified impacts of patterns on soft goals, which are derived directly from the ontology and weighted by the user's propagated preferences. This evaluation matrix will be updated in the next phases.

$$M = \begin{bmatrix} +1 & 0 & -1 \\ +1 & +1 & -1 \\ -1 & -1 & +1 \end{bmatrix}$$

For illustration, by considering the current example, where the three proposed patterns on the Soft goal Map (Off-Chain Storage, Centralized Oracle, and Decentralized Oracle) evaluated against three soft goals (Cost Reduction, Performance Improvement, and Security Enhancement). After translating qualitative impacts (improves, worsens, neutral) into numerical values (+1, –1, 0) and applying stakeholder preferences as

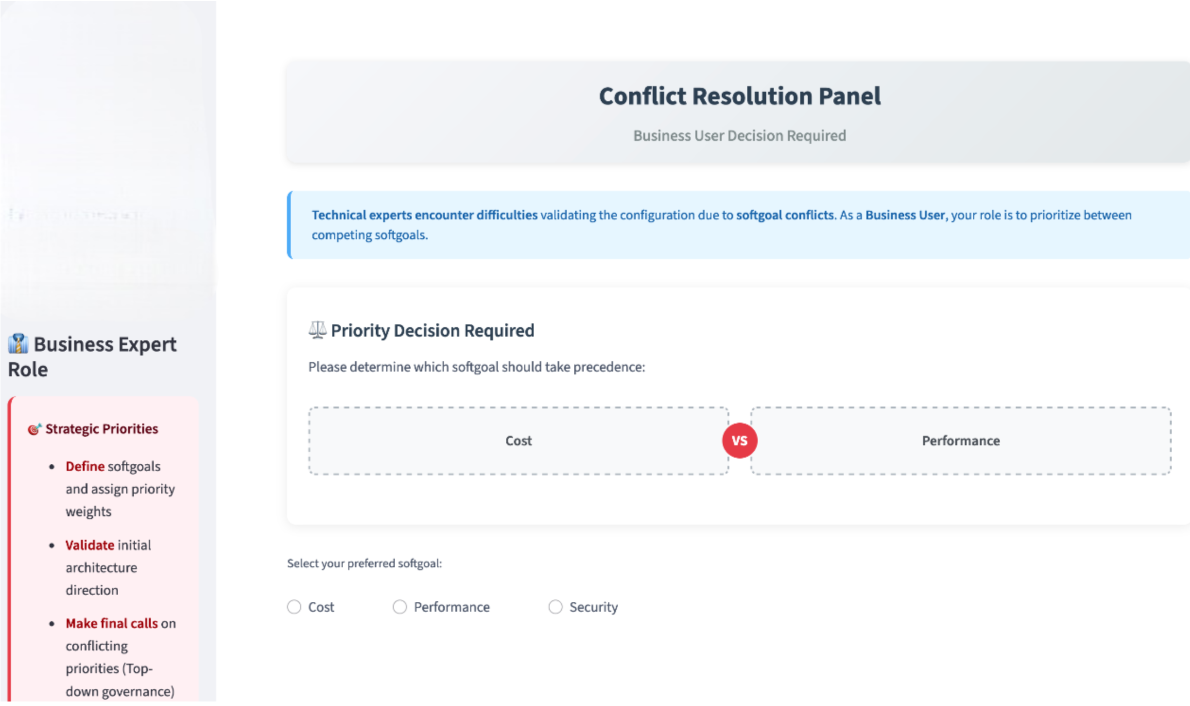

FIGURE 7.7: Conflict Resolution and Trade-offs Analysis

weights.

Once conflicts are resolved and stakeholder priorities are clearly defined, the BC-TEAEM proceeds to re-generate Softgoal Map. This map represents the set of blockchain patterns that best align with the prioritized input soft goals and makes explicit how each pattern contributes to the different soft goal.

Softgoal Map: In the illustrative use case, the business users assign the higher priorities to *performance*. Based on this preference, the recommendation engine queries the knowledge base and construct the configuration that highlights three main patterns: *the Centralized Oracle*, which facilitates communication with external systems; *the Decentralized Oracle*; and the *Off-Chain Data Storage with Cryptographic Proof*. These patterns are proposed because they collectively address the prioritized performance goals, while still providing varying degrees of support or compromise for the other soft goals.

Visualization Encoding: To facilitate interpretation, the engine employs a visual encoding scheme that translates the ontology's numerical representation of patter-soft goal impacts into intuitive graphical indicators.

- *A full green bar* represents a strong positive contribution (+2 as encoded in the ontology),
- *a semi-filled green bar* indicates moderate positive influence (+1),
- *a red bar* denotes a negative influence (–1 or –2), and
- *a gray bar* reflects a neutral impact (0).

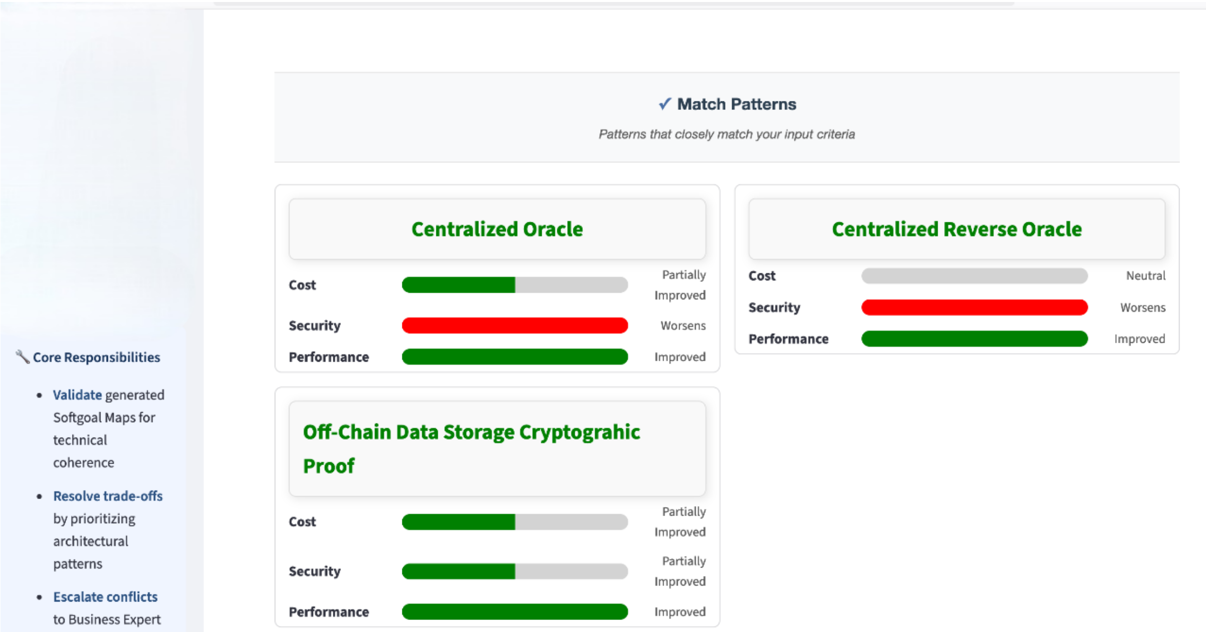

FIGURE 7.8: Final Soft goal Map proposed by the BC-TEAEM

This graphical representation allows stakeholders to quickly grasp how each pattern aligns with their prioritized objectives. Once the visualization is reviewed, the technical users can proceed to validate the first configuration, confirming that the proposed set of patterns is both technically feasible and consistent with the business priorities.

After reviewing the generated Soft goal Map and its visual representation, the process transitions to the technical users. Their role is to validate the proposed configuration by assessing its technical feasibility and aligning them with system constraints. Validation at this stage ensures that the recommended patterns are not only aligned with business priorities but are also practically implementable.
Integration of Non-Functional Patterns: After the validation of the initial configuration, the BC-TEAEM proceeds to a complementary step in which non-functional patterns are proposed. These patterns are displayed as interactive cards, each accompanied by a score computed through the TOPSIS Algorithm. The score reflects the extent to which the non-functional pattern can potentially improve the satisfaction of the prioritized soft goals and, the overall trust score of the system.

The rationale of this step is that, while the initial configuration provides a baseline alignment with business users priorities, additional non-functional patterns can reinforce system-level qualities such as scalability, privacy, or auditability. These qualities may not have been explicitly requested at the outset but can significantly enhance the robustness and credibility of the final solution.

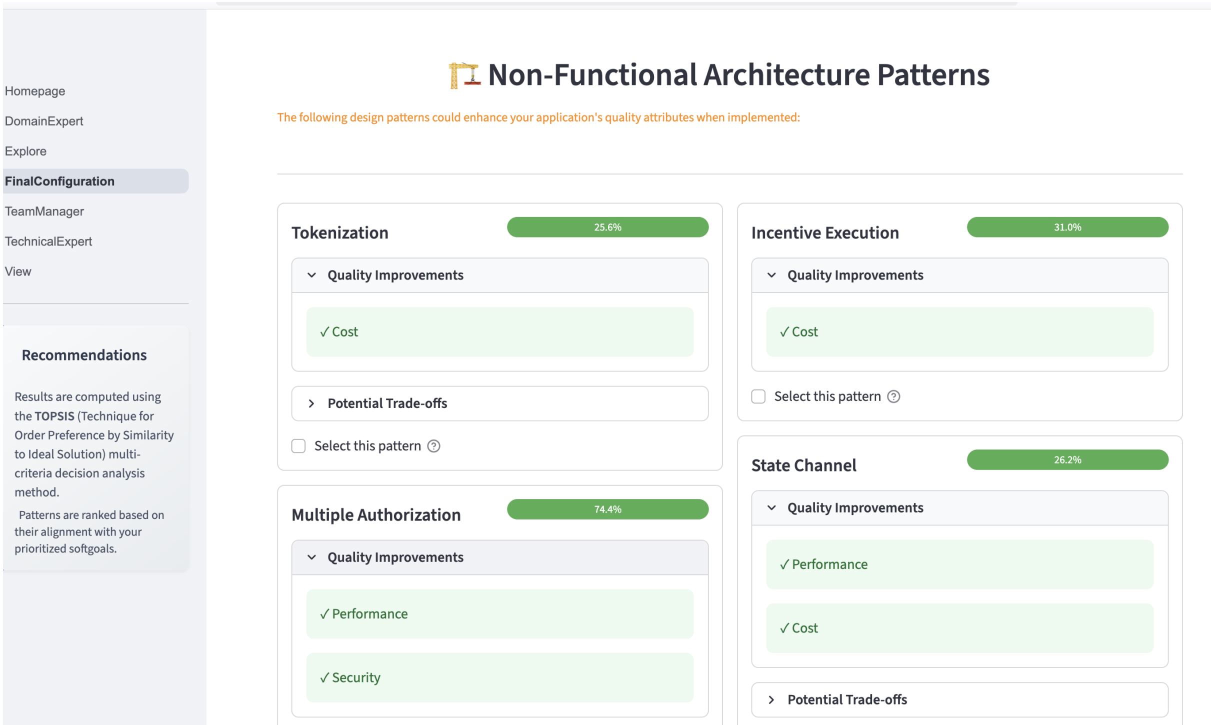

FIGURE 7.9: Adding Non-Functional Patterns

At this stage, the technical users are empowered to review the set of suggested non-functional patterns and decide whether to include or exclude them from the configuration. This flexibility allows technical experts to tailor the recommendation to practical implementation constraints, organizational policies, or regulatory requirements.
Once the selection is finalized, BC-TEAEM produces the final validated configuration, which integrates both the functional and non-functional patterns chosen. This configuration represents the outcome of the iterative decision-making process, incorporating stakeholder preferences, conflict resolution decisions, and technical validations. This configuration can then be exported for documentation, implementation planning, or further evaluation in the system design process.
The export mechanism serves several purposes:

- **Documentation:** The final configuration is exported in a structured format (JSON Format), ensuring that the decision-making process and its outcomes are recorded.
- **Traceability:** Alongside the selected patterns, the exported configuration includes the associated softgoal map, the propagated preference weights, and the scores computed by TOPSIS. This allows future reviewers to trace back how and why a given decision was made, reinforcing transparency.
- **Implementation Guidance:** The exported configuration can be used as a blueprint for system architects and developers. By providing a consolidated view of both functional and non-functional patterns, it bridges the gap between high-level

decision support and technical system design.

- **Reuse and Iteration:** The exported configuration may also serve as input for subsequent iterations of the decision process. For example, in a later project phase, stakeholders may revisit the configuration, adjust priorities, or extend the set of considered patterns.

In this way, the export function ensures that BC-TEAEM does not stop at delivering recommendations but contributes directly to the operationalization of trust-oriented blockchain solutions. The final configuration becomes a tangible artifact that can guide implementation, facilitate collaboration, and support compliance verification.

7.7 Evaluation

To assess the validity of our approach, we combine two complementary evaluation strategies. The first is a conceptual validation using Structural Equation Modeling (SEM) to test our theoretical soundness of the nomological network of trust. The second is an operational validation, where we instantiate our approach into the BC-TEAEM tool and assess its ability to guide design decision in practice in order to enhance trust.

7.7.1 Operational validation via BC-TEAEM

Following design science principles, we evaluated BC-TEAEM using a mixed-methods approach that combines:

- Scenario-based experimentation with domain-relevant tasks to assess recommendation relevance and trade-off clarity.
- Structured questionnaires to measure perceived usability, usefulness, and user confidence.

This dual method was chosen to capture both qualitative insights (e.g., expert reasoning, observed interactions) and quantitative perceptions (e.g., Likert-Scale responses).
To guide our evaluation, we formulated the following hypotheses:

H_1: *The tool supports decision-making by providing relevant recommendations and clear trade-offs when used in realistic design scenarios.*

H_2: *The tool is perceived as usable, useful, and understandable by users, including those with limited experience in blockchain pattern selection.*

Scenario-based experimentation

Purpose: Evaluate whether BC-TEAEM produces relevant recommendations and clearly communicates trade-offs in realistic design scenarios (addresses RQ_1 and RQ_2 and H_1).

Data Collection: The evaluation involved seven researchers with blockchain and/or software design experience (Table 7.3).

TABLE 7.3: Participant Panel overview

	E1	E2	E3	E4	E5	E6	E7
Role	R1	R2	R3	R4	R5	R6	R7
Blockchain Exp (y)	4	4	4	4	2	1	2
Software Design Exp	5+	1	5	5	2	2	5+

Experimental Settings: Each participant was presented with a scenario involving a logistics enterprise seeking to prioritize three soft goals: *cost reduction, performance improvement, and security enhancement*. They used the tool to generate pattern recommendations and evaluate trade-offs between variants (e.g., Oracle vs. Reverse Oracle). Assessing the tool's output for relevance, clarity of trade-offs, and interface usability.

Data Analysis & Results: The data collected from the semi-structured interviews and tool interactions (logs). The results of the scenario-based evaluation showed that the tool successfully recommended patterns aligned with the specified soft goals. The tool identified **Off-Chain Data Storage** as the most suitable pattern, and highlighted trade-offs between Oracle and Reverse Oracle pattern variants. Responses were categorized into:

- **Relevance of Recommendations:** Five of seven participants agreed aligned with industry practices (e.g., Off-Chain Data Storage). Minor disagreements (E2, E6) highlighted contextual nuances in cost-performance trade-offs.
- **Clarity of Trade-offs:** 6/7 found them "clearly presented and comprehensive". E1 noted that additional examples could further improve clarity.
- **Usability of the Interface:** 4 experts found the interface intuitive for a 30-minute session, while 3 suggested streamlining terminology (e.g., "Reverse Oracle" required glossary support).

Threats to Validity.

- **External:** Limited to supply chain scenarios; generalization requires broader domains.
- **Construct:** Time constraints (30 minutes) may not reflect real-world deliberation.

Structured Questionnaire

Purpose: To evaluate $\mathbf{H}_2$ and textbfRQ$_3$—the perceived usability, usefulness, and understandability of the tool by a broader audience, including less experienced users.

Data Collection: we collected data from 32 participants (26 postgraduate students, 4 researchers, and 2 industry experts). The Structured Google Forms questionnaire was based on TAM (Marangunić and Granić, 2015) and SUS (Vlachogianni and Tselios, 2022), assessing:

- *Usability* which evaluates how easy and intuitive the tool is to use.
- *Self-efficacy* measuring users' confidence in independently using the tool to achieve their goals (for instance, exploring blockchain patterns), and
- *Usefulness of Features* assessing whether the tool's functionalities (recommendations, pattern comparisons) meet users' needs.

Experimental Protocol: Participants interacted with the BC-TEAEM tool during a supervised lab session exploring pattern recommendations for a given set of soft goals. Participants rated the tool's usability, self-efficacy, and usefulness of features.

Data Analysis & Results: Descriptive statistics were computed from responses (Table 7.4). This analysis helped us understand the overall perception of the tool and identify areas for improvement.

TABLE 7.4: Perception Scores from Participants

Category	Average/5	Standard Deviation
Usability	3.9	1.1
Self-Efficacy	3.4	1.2
Usefulness of Features	4.1	0.8

The results indicate that the tool is perceived as highly useful (average of 4.1/5), with its blockchain pattern recommendations considered valuable for application design. However, usability (3.9/5) and self-efficacy (3.4/5) scores reveal challenges in clarity and user confidence. The higher standard deviations for these categories reflect divergent user experiences—some found the tool intuitive, while others highlighted the need for additional guidance (e.g., video tutorials, tooltips) or training. Feedback suggests prioritizing improvements such as clearer feature explanations and a more intuitive interface. Implementing these adjustments could enhance adoption by reducing the learning curve.

Threats to Validity:

- **Sampling bias:** The sample over-represents students (87%), which may limit direct generalization to industry practice. However, their diversity in technical backgrounds provided varied perspectives on usability and clarity.

7.8 Threats to Validity

7.8.1 Construct Validity:

The Blockchain domain have an undefined number of blockchain patterns. The selected patterns and soft goals in this work may not exhaust the domain. To ensure the ontology remains relevant and comprehensive, a systematic review of emerging blockchain design patterns should be conducted. This would allow the continuous integration of the state-of-the-art patterns into the ontology, thus maintaining its accuracy and applicability in evolving domains.

7.8.2 Internal validity

Contribution polarities and intensities partly rely on literature synthesis and expert judgment, which may introduce bias.
Mitigation: (i) triangulation across multiple sources; (ii) recording provenance at the Contribution level; (iii) sensitivity analyzes in the MCDM step to test the robustness of recommendations against perturbations of intensity values

7.8.3 External validity

The ontology was developed with inter-organizational processes and Supply Chain scenarios in mind. Its applicability to other domains such as healthcare DeFi, requires empirical assessment.
Mitigation: we favored general OntoUML structures (relators for influence, roles for contextualized participation) to maximize portability.

7.8.4 Conclusion validity

While the ontology structures reasoning, practical effectiveness depends on downstream modules (e.g., TOPSIS configuration, stakeholder weights).
Mitigation: we report decision paths (traceable Contributions) and include alternative weight settings to detect over-fitting to a single preference profile.

7.9 Related Work

The research area on blockchain pattern recommendation has been evolving with several contributions focusing on cataloging, structuring, and providing partial decision support to practitioners. However, a closer examination reveals that existing approaches often remain limited in their scope, either by focusing narrowly on ontological descriptions of patterns or by relying on static decision- making models that capture user preferences and trade-offs in insufficient ways. In contrast, our BC-TEAEM framework seeks to advance this field by integrating pattern ontologies, explicit user preferences, and multi-criteria decision support within a single interactive tool.

Several works have emphasized the need to structure blockchain patterns into formal ontologies to facilitate their reuse and analysis. For example, papers in (Six, Correa-Restrepo, et al., 2022; Six, Herbaut, and Salinesi, 2022a; Xu et al., 2018; Xu et al., 2025) develop ontologies that represent blockchain patterns, their intrinsic features, and their potential applications. These ontologies provide a valuable foundation for reasoning about patterns. Nonetheless, they primarily remain descriptive: while they organize patterns systematically, they do not establish explicit links between patterns and soft goals such as performance, security, or scalability. BC-TEAEM extend this line of work by not only reusing and enriching existing ontologies but also embedding soft goals into the representation. This integration allows design decisions to be evaluated not only structurally but also in terms of their alignment with user-defined priorities.

Another research direction has focused on the integration user needs into pattern recommendation. For instance, authors in (Soundararajan and Shenbagaraman, 2023 and Bahar et al., 2023) highlight the importance of adapting recommendations to contextual and user-specific requirements. However, these approaches often stop short of offering explicit mechanisms for weighting soft goals systematically. BC-TEAEM builds on this insight by introducing an explicit weighting system, where users can directly express the relative importance of different soft goals. This operationalization ensures that recommendations are not generic but tailored to the decision-maker's strategic priorities.

For decision-support methods and tools, existing approaches exist in the literature such as proposed by (Six, Herbaut, and Salinesi, 2022a and Liu et al., 2022) that employ the multi-criteria decision analysis (MCDA) techniques like TOPSIS. Although these methods allow comparison of alternatives, they are based on static attributes (features) that fail to reflect interdependencies among soft goals and the dynamic influence patterns exert on one another. BC-TEAEM overcomes this limitations by combining an ontology of blockchain features, patterns and soft goals with the influence matrix of weight, and multi-criteria decision-support (TOPSIS); thus capturing the systemic interactions and trade-offs inherent in blockchain design decisions.

The issue of trade-offs has been acknowledged but often remains implicit or underdeveloped in the literature. (Jovanovic et al., 2022) address the challenge of managing trade-offs in blockchain adoption but provide no structured mechanism to make them explicit during pattern selection. Our approach explicitly models these trade-offs by linking positive and negative influences between patterns and soft goals. This makes trade-offs not only visible but also actionable within the decision-support process.

Finally, with respect to tool support, existing contributions are often conceptual or remain limited to prototypes of reduced scope. (Six, Correa-Restrepo, et al., 2022), for example propose ontological prototypes, but these primarily demonstrate feasibility rather than providing interactive decision-making functionalities. In contrast, BC-TEAEM is implemented as an interactive prototype that enables dynamic recommendations, allowing practitioners to explore scenario, adjust soft goal weights, and immediately observe their impact on pattern selection. This feature transforms pattern recommendation from a static lookup process into a guided, iterative decision-making experience.

In summary, the reviewed works each address isolated aspects of the blockchain pattern recommendation problem: ontological structuring, user contextualization, decision-support, or trade-off awareness. Yet, the absence of a unified framework limits their applicability in real-world settings where all these aspects intersect. BC-TEAEM fills this research gap by providing an integrated system that connects ontologies, user preferences, and dynamic decision support into a coherent methodology, thereby advancing the state of the art in blockchain pattern recommendation.

Contribution Dimension	**Prior Work Example**	**Limitation**	**BC-TEAEM Contribution**
Ontology of Patterns	Six, Correa-Restrepo, et al., 2022; Xu et al., 2018; Six, Correa-Restrepo, et al., 2022	Pattern ontologies without soft goal links	Reused and extended ontology by integrating soft goals.
User Preferences	Soundararajan and Shenbagaraman, 2023; Bahar et al., 2023	No explicit weighting of soft goals	Explicit soft goal weighting by the user.
Decision Support	Six, Herbaut, and Salinesi, 2022a; Liu et al., 2022	TOPSIS based on static attributes	TOPSIS combined with soft goal influence matrix.
Trade-off Management	Jovanovic et al., 2022	Implicit or missing	Explicit modeling of trade-offs between patterns.
Tool Support	Six, Correa-Restrepo, et al., 2022	Conceptual tools or limited prototypes	Interactive prototype enabling dynamic recommendations.

TABLE 7.5: Analytical Comparison with Existing Approaches

7.10 Conclusion

This Chapter presented the validation of an OntoUML-based ontology for modeling blockchain software patterns and trust-related soft goals. The ontology satisfies internal validity criteria by conforming to OntoUML/UFO commitments, maintaining well-formed taxonomies, and using relators to ground pattern–soft goal influences. Relevance was established through competency questions, traceability to curated sources, and expert reviews on representative scenarios. Instantiation experiments confirmed that the ontology supports explainable trade-off analysis, exposing the exact contribution structures that drive recommendations.

Acknowledging residual threats, the ontology nonetheless offers a robust foundation for integrating conceptual rigor with decision support. It underpins the subsequent components of the BC-TEAEM approach, where stakeholder preferences and multi-criteria methods (e.g., TOPSIS) operate over well-founded contribution structures to recommend patterns under trust constraints. In the next chapter, we capitalize on this foundation to implement preference-aware reasoning and to evaluate the approach on practical case studies.

Chapter 8

General Conclusion and Perspectives

8.1 Conclusion

This thesis addressed the longstanding problem of trust in inter-organisational business processes, where heterogeneous stakeholders must coordinate activities, exchange data, and reach common decisions without necessarily sharing governance structures, risk profiles, Information Systems, or incentives. Existing approaches in the literature, and in practice, ranging from classical enterprise architecture to blockchain-based applications offer partial support but lack a unified mechanism for connecting business-level trust requirements with concrete technical design choices in the systematic manner. This gap motivated the design and development of BC-TEAEM, a framework that integrates conceptual modeling, ontology engineering, and multi-criteria decision support to guide blockchain pattern selection under trust constraints.

The research unfolded across several research streams. By following the DSR cycle, we first do the ***Problem identification phase:*** We began by defining and clarifying the concept of trust in IOBPs. We categorize three types of trust; social, technological and digital that we use for the identification of recurring trust issues that arise prior collaboration. Through a scoping review, we made a taxonomy of these trust issues into seven groups in order to transform that into an explicit trust requirements that will be exploited. Then building on this foundation, we investigated how blockchain technology can contribute to mitigating these trust issues. Rather than treating blockchain as a monolithic solution, we analyzed its intrinsic features and patterns, and systematically mapped them to identified trust requirements. This mapping highlighted both the potential benefits and the inherent trade-offs of blockchain adoption in inter-organizational processes.

For the second phase of DSR, ***Treatment Design:*** To design an artifact that align between organizational goals (trust or not) and technical decisions, we developed TEAEM: Technology–Aware Enterprise Modeling. TEAEM extends the Model-Driven Architecture paradigm by integrating bottom-up constraint propagation and analysis allowing technical properties to influence higher-level enterprise models. This bidirectional reasoning enables a more realistic and justifiable alignment between business goals and technological choices.

We further demonstrated the applicability of TEAEM by integrating it with Software Product Line Engineering. This integration enables the generation and evaluation of alternative configurations that satisfy trust-related soft goals such as transparency, security, and performance; while accounting for stakeholder preferences and technical constraints. By doing so, TEAEM provides a systematic means to reason about trust-aware configurations across heterogeneous inter-organizational settings.

For the last phase of DSR, ***Treatment Validation:*** We operationalize TEAEM with a framework that we call BC-TEAEM. BC-TEAEM combines an ontology of blockchain patterns and trust-related soft goals with a multi-criteria decision analysis engine to recommend suitable blockchain patterns based on specified trust requirements and stakeholder preferences. The framework captures complex interdependencies among patterns, soft goals, and trade-offs, enabling systematic reasoning about design choices in trust-sensitive contexts. We validated BC-TEAEM through a prototype implementation and demonstrated its utility in guiding blockchain pattern selection for trust-sensitive inter-organizational processes, such as supply chain management. The results highlight that BC-TEAEM effectively supports architects in navigating the complex landscape of trust-related design decisions. The framework thus advances both research and practice by providing a traceable bridge between high-level trust requirements and blockchain architectural decisions.

Overall, the thesis contributes a holistic approach to trust-aware design in inter-organizational systems. By integrating ontological modeling and multi-criteria analytics, BC-TEAEM responds to the fragmentation of existing approaches and demonstrates how trust can be treated not only as a behavioral or organizational problem but also as a technical design concern supported by formal reasoning.

In summary, this research advances the state-of-the-art in trust-aware enterprise modeling by providing a systematic framework that connects business-level trust requirements with concrete blockchain design choices. The contributions span conceptual, methodological, and practical dimensions, offering a comprehensive solution to the challenges of trust in inter-organizational business processes.

8.2 Limitations

Despite its strong contributions, this research remains subject to some limitations that should be acknowledged. These limitations concern conceptual scope, methodological constraints, and implementation-related aspects.

First, trust is broader than what can be captured by soft-goals and blockchain mechanisms. Behavioral trust, legal trust, organizational governance, and human factors remain only partially represented. While the thesis positions blockchain as an enabler of structural trust, it does not fully model social dynamics such as incentives, compliance culture, or inter-organizational negotiation. These factors can significantly affect the success of trust-enhancing technologies but lie beyond the scope of the current work.

Second, the ontology does not cover all possible blockchain patterns or trust-related soft goals. Although the BC-TEAEM knowledge base consolidates the state-of-the-art pattern catalogs, blockchain technology evolves rapidly. Emerging paradigms—such as modular blockchains, zero-knowledge, cross-chain protocols, and verifiable off-chain computation are not yet integrated. The framework is designed to be extensible, but the current version still reflects a bounded set of design options.

Third, the representation of soft-goal influence networks is necessarily simplified. Relations between patterns and soft goals were modeled qualitatively (positive, negative, neutral), which supports explainability but limits granularity. In real-world systems, influences may be non-linear, conditional, or context-dependent. For example, privacy does not uniformly reduce transparency, the effect depends on the cryptographic techniques used. More fine-grained quantitative modeling could enhance precision but would require detailed empirical data not currently available.

Fourth, the evaluation of pattern impacts relies partly on expert judgment. Although grounded in literature synthesis and validated by domain experts, the contribution scores still reflect subjective assessments. This is a common limitation in goal-oriented requirements engineering and multi-criteria decision-making. Stakeholders may disagree on the magnitude of impacts, particularly in domains where empirical performance data are scarce or implementation architectures vary significantly. The engine mitigates this issue by providing explainability and allowing iterative adjustment, but the underlying subjectivity remains.

Fifth, the validation of the framework is limited to qualitative and semi-structured evaluations. Ontological validation includes expert review, instantiation experiments, and consistency checks, but lacks large-scale empirical deployment. The case studies demonstrate feasibility and usefulness, but not statistical generalizability. Additional real-world evaluations across multiple industries, organizational contexts, and technical stacks would strengthen the external validity of BC-TEAEM.

8.3 Future Works

The different research contributions developed in this thesis open several perspectives for the future research. These perspectives are structured according to the main artifacts developed in this work, namely the scoping review on trust and blockchain, the TEAEM framework, and the BC-TEAEM framework.

Extending the Scoping Review on Trust and Blockchain

The scoping review conducted in Chapter 4 provides a foundational understanding of trust issues in inter-organizational business processes and how blockchain technology can address them. Future extensions can be envisioned.

The review could be extended both temporally and thematically. Blockchain domains and trust-related challenges evolve rapidly, and new research directions continuously emerging. Periodic updates to the review would ensure that it remains current and captures emerging trends. Thematically, future reviews could explore specific sectors (e.g., healthcare, finance) or emerging blockchain paradigms (e.g., DeFi, NFTs) to identify domain-specific trust issues and solutions.

Additionally, the taxonomy of trust issues could be validated and refined through empirical studies involving practitioners. Surveys, interviews, or case studies could provide real-world insights into how organizations perceive, strengthen the external validity of the identified trust dimensions and help capture sector-specific trust issues that are not fully represented in the literature. This complement the literature-based taxonomy.

Technology-Aware Enterprise Modeling (TEAEM)

Several research directions can further enhance the Technology-Aware Enterprise Modeling (TEAEM) approach proposed in this thesis.

TEAEM could be extended to support a boarder range of technological domains beyond blockchain. While this thesis focuses on blockchain technologies as trust-enabling technology, the underlying principles of bottom-up constraint propagation and bidirectional alignment between business goals and technical choices are applicable to other paradigms. Future work could explore how TEAEM can integrate with cloud computing, Internet of Things (IoT), or artificial intelligence (AI) technologies, each of which introduces its own set of technical constraints and trust considerations.

The integration of TEAEM with existing enterprise architecture frameworks and model-driven engineering tools represents an important perspective. Developing plugins or extensions for popular EA tools (e.g., ArchiMate, TOGAF) would facilitate adoption by practitioners and enable seamless integration into existing modeling workflows. This would also allow for more comprehensive evaluations of TEAEM in real-world enterprise settings.

Also, empirical validation of TEAEM through case studies and industrial collaborations would strengthen its practical relevance. Future research could involve deploying TEAEM in real organizational contexts to assess its effectiveness in aligning business goals with technical decisions. Such studies could provide insights into usability, scalability, and the impact of TEAEM on decision-making processes.

Blockchain Technology-Aware Enterprise Modeling (BC-TEAEM)

The BC-TEAEM framework constitutes a central contribution of this thesis and offers several promising directions for future research

First, the BC-TEAM knowledge base could be expanded to include additional blockchain patterns and more granular trust-related soft goals. As blockchain technology continues to evolve, new design patterns and trust-related requirements will emerge. Integrating emerging paradigms such as cross-chain interoperability, layer-2 scaling solutions, and privacy-enhancing technologies would enhance the coverage and relevance of the framework.

Second, the influence modeling between patterns and soft goals could be refined to incorporate more quantitative and empirical evidence. Complementing expert-based impact assessments with performance benchmarks, formal security metrics, and real-world case studies would reduce subjectivity and improve the reliability of the evaluation process.

Third, the recommendation engine could be enhanced with advanced decision-making techniques. While TOPSIS provides a solid foundation for multi-criteria analysis, exploring alternative methods such as Analytic Hierarchy Process (AHP), ELECTRE, or machine learning-based approaches could offer different trade-offs in terms of explainability, scalability, and adaptability to complex decision contexts.

A particularly promising future direction concerns the integration of Artificial Intelligence techniques into BC-TEAM. Machine learning algorithms could be employed to analyze historical data on blockchain deployments and trust outcomes, enabling the system to learn patterns and improve recommendations over time. Large Language Models (LLMs) could be leveraged to assist in knowledge evolution, pattern identification, and automated extraction of trust-related requirements from unstructured data sources (technical or scientific documentation). Causal AI models could also be leveraged to simulate how architectural decisions influence trust-related outcomes, thereby strengthening the explainability of recommendations.

Finally, large-scale and cross-domain validation will be essential for advancing BC-TEAEM. Applying the framework across varied sectors—such as finance, healthcare, and energy—will assess its generalizability, robustness, and practical utility while further refining its trust-oriented decision-support capabilities.

> **In conclusion, this thesis demonstrates that trust in inter-organizational systems can be approached as a design problem, grounded in explicit modeling, traceable reasoning, and informed technological choices. By bridging conceptual foundations and operational decision support, BC-TEAEM contributes to a more systematic and transparent engineering of trustworthy collaborative systems.**

Bibliography

Aalst, W. (2013). Business process management: A comprehensive survey. ISRN Software Engineering, 2013(1), 1–37.

Akrout, H. (2015). A process perspective on trust in buyer–supplier relationships."calculus" an intrinsic component of trust evolution. European Business Review, 27(1), 17–33.

Aleisa, M. A. (2025). Blockchain-enabled zero trust architecture for privacy-preserving cybersecurity in iot environments. IEEE Access.

Alves, P. H., Paskin, R., Frajhof, I., Miranda, Y., Jardim, J. G., Cardoso, J., Tress, E., Cunha, R., Nasser, R., & Carvalho, G. (2020). Exploring blockchain technology to improve multi-party relationship in business process management systems. https://doi.org/10.5220/0009565108170825

Arksey, H., & O'malley, L. (2005). Scoping studies: Towards a methodological framework. International journal of social research methodology, 8(1), 19–32.

Bachmann, R., & Zaheer, A. (2013). Handbook of advances in trust research. Edward Elgar Publishing.

Bagheri, E., Ensan, F., Gasevic, D., & Boskovic, M. (2011). Modular feature models: Representation and configuration. Journal of Research and Practice in Information Technology, 43(2), 109–140.

Bahar, M. N., Ries, M., & Košt'ál, K. (2023). A dynamic decision support system for the best-fitting blockchain platform selection. 2023 Fifth International Conference on Blockchain Computing and Applications (BCCA), 412–419.

Bapatla, A. K., Mohanty, S. P., Kougianos, E., Puthal, D., & Bapatla, A. (2023). Pharmachain: A blockchain to ensure counterfeit-free pharmaceutical supply chain. IET Networks, 12(2), 53–76.

Baralla, G., Pinna, A., Tonelli, R., Marchesi, M., & Ibba, S. (2021). Ensuring transparency and traceability of food local products: A blockchain application to a smart tourism region. Concurrency and Computation: Practice and Experience, 33(1), e5857.

Bedin, A. R. C., Capretz, M. A. M., & Mir, S. (2021). Blockchain for collaborative businesses. Mobile Networks and Applications, 26, 277–284. https://doi.org/10.1007/s11036-020-01649-6

Belotti, M., Božić, N., Pujolle, G., & Secci, S. (2019). A vademecum on blockchain technologies: When, which, and how. IEEE Communications Surveys & Tutorials, 21(4), 3796–3838. https://doi.org/10.1109/COMST.2019.2928178

Besancon, L., Da Silva, C. F., Ghodous, P., & Gelas, J.-P. (2022). A blockchain ontology for dapps development. IEEE Access, 10, 49905–49933.

Biswas, D., Jalali, H., Ansaripoor, A. H., & De Giovanni, P. (2023). Traceability vs sustainability in supply chains: The implications of blockchain. European Journal of Operational Research, 305(1), 128–147.

Blanchet, B. (2016). Modeling and verifying security protocols with proverif. Foundations and Trends in Privacy and Security, 1(1-2), 1–135.

Bošković, M., Mussbacher, G., Bagheri, E., Amyot, D., Gašević, D., & Hatala, M. (2011). Aspect-oriented feature models. Models in Software Engineering: Workshops and Symposia at MODELS 2010, Oslo, Norway, 110–124.

Bouchbout, K., & Alimazighi, Z. (2011). Inter-organizational business processes modelling framework. ADBIS.

Braga, D. D. S., Niemann, M., Hellingrath, B., & Neto, F. B. D. L. (2018). Survey on computational trust and reputation models. ACM Computing Surveys (CSUR), 51(5), 1–40.

Buschmann, F., Meunier, R., Rohnert, H., Sommerlad, P., & Stal, M. (1996). Software patterns.

Buterin, V., et al. (2014). A next-generation smart contract and decentralized application platform. white paper, 3(37), 2–1.

Caro, M. P., Ali, M. S., Vecchio, M., & Giaffreda, R. (2018). Blockchain-based traceability in agri-food supply chain management: A practical implementation. 2018 IoT Vertical and Topical Summit on Agriculture - Tuscany (IOT Tuscany), 1–4. https://doi.org/10.1109/IOT-TUSCANY.2018.8373021

Casino, F., Kanakaris, V., Dasaklis, T. K., Moschuris, S., & Rachaniotis, N. P. (2019). Modeling food supply chain traceability based on blockchain technology [9th IFAC Conference on Manufacturing Modelling, Management and Control MIM 2019]. IFAC-PapersOnLine, 52(13), 2728–2733. https://doi.org/https://doi.org/10.1016/j.ifacol.2019.11.620

Casino, F., Kanakaris, V., Dasaklis, T. K., Moschuris, S., Stachtiaris, S., Pagoni, M., & Rachaniotis, N. P. (2021). Blockchain-based food supply chain traceability: A case study in the dairy sector. International Journal of Production Research, 59(19), 5758–5770. https://doi.org/10.1080/00207543.2020.1789238

Chang, S. E., & Chen, Y. (2020). When blockchain meets supply chain: A systematic literature review on current development and potential applications. IEEE Access, 8, 62478–62494. https://doi.org/10.1109/ACCESS.2020.2983601

Chopra, K., & Wallace, W. A. (2003a). Trust in electronic environments. 36th Annual Hawaii International Conference on System Sciences, 2003. Proceedings of the, 10–pp.

Chopra, K., & Wallace, W. A. (2003b). Trust in electronic environments. 36th Annual Hawaii International Conference on System Sciences, 2003. Proceedings of the, 10–pp.

Clarke, E. M., Grumberg, O., & Peled, D. (1999). Model checking. MIT Press.

Cooper, D., Santesson, S., Farrell, S., Boeyen, S., Housley, R., & Polk, W. (2008). Internet x.509 public key infrastructure certificate and certificate revocation list (crl) profile. https://datatracker.ietf.org/doc/html/rfc5280

Cortes-Cornax, M., Matei, A., Dupuy-Chessa, S., Rieu, D., Mandran, N., & Letier, E. (2015). Using intentional fragments to bridge the gap between organizational and intentional levels. Information and Software Technology, 58, 1–19.

De Giovanni, P. (2020). Blockchain and smart contracts in supply chain management: A game theoretic model. International Journal of Production Economics, 228, 107855.

de la Vara, J. L., Sánchez, J., & Pastor, O. (2013). On the use of goal models and business process models for elicitation of system requirements. International Workshop on Business Process Modeling, Development and Support, 168–183.

Di Ciccio, C., Cecconi, A., Dumas, M., García-Bañuelos, L., López-Pintado, O., Lu, Q., Mendling, J., Ponomarev, A., Binh Tran, A., & Weber, I. (2019). Blockchain support for collaborative business processes. Informatik Spektrum, 42, 182–190.

Di Ciccio, C., Cecconi, A., Mendling, J., Felix, D., Haas, D., Lilek, D., Riel, F., Rumpl, A., & Uhlig, P. (2018). Blockchain-based traceability of inter-organisational business processes. International Symposium on Business Modeling and Software Design, 56–68.

documentation, I. (2018). Iso/iec/ieee international standard - systems and software engineering – life cycle processes – requirements engineering. ISO/IEC/IEEE 29148:2018(E), 1–104. https://doi.org/10.1109/IEEESTD.2018.8559686

Dong, S., Abbas, K., Li, M., & Kamruzzaman, J. (2023). Blockchain technology and application: An overview. PeerJ Computer Science, 9, e1705.

Dumas, M., Rosa, L. M., Mendling, J., & Reijers, A. H. (2018). Fundamentals of business process management. Springer.

Durugbo, C. M., & Al-Balushi, Z. (2023). Supply chain management in times of crisis: A systematic review. Management Review Quarterly, 73(3), 1179–1235.

Figorilli, S., Antonucci, F., Costa, C., Pallottino, F., Raso, L., Castiglione, M., Pinci, E., Del Vecchio, D., Colle, G., Proto, A. R., Sperandio, G., & Menesatti, P. (2018). A blockchain implementation prototype for the electronic open source traceability of wood along the whole supply chain. Sensors, 18(9).

Franch, X., López, L., Cares, C., & Colomer, D. (2016). The i* framework for goal-oriented modeling. Domain-Specific Conceptual Modeling: Concepts, Methods and Tools, 485–506.

Gambetta, D., et al. (2000). Can we trust trust. Trust: Making and breaking cooperative relations, 13(2000), 213–237.

Garcia-Garcia, J. A., Sánchez-Gómez, N., Lizcano, D., Escalona, M. J., & Wojdyński, T. (2020). Using blockchain to improve collaborative business process management: Systematic literature review. IEEE Access, 8, 142312–142336. https://doi.org/10.1109/ACCESS.2020.3013911

Gazi, Y., Umar, M. S., & Sadiq, M. (2015). Classification of nfrs for information system. International Journal of Computer Applications, 115(22).

Giannakis, M., Croom, S., & Slack, N. (2004). Supply chain paradigms. Understanding supply chains, 1–22.

Gräbner, C., Elsner, W., & Lascaux, A. (2021). Trust and social control: Sources of cooperation, performance, and stability in informal value transfer systems. Computational Economics, 58(4), 1077–1102. https://doi.org/10.1007/s10614-020-09994-0

Gröner, G., Asadi, M., Mohabbati, B., Gašević, D., Bošković, M., & Parreiras, F. S. (2014). Validation of user intentions in process orchestration and choreography. Information Systems, 43, 83–99.

Guizzardi, G., Wagner, G., Almeida, J. P. A., & Guizzardi, R. S. (2015). Towards ontological foundations for conceptual modeling: The unified foundational ontology (ufo) story. Applied ontology, 10(3-4), 259–271.

Guizzardi, R., & Reis, A. N. (2015). A method to align goals and business processes. International Conference on Conceptual Modeling, 79–93.

Hameed, H., Zafar, N. A., Alkhammash, E. H., & Hadjouni, M. (2022). Blockchain-based formal model for food supply chain management system using vdm-sl. Sustainability, 14(21). https://doi.org/10.3390/su142114202

Hasan, H. R., Salah, K., Jayaraman, R., Ahmad, R. W., Yaqoob, I., & Omar, M. (2020). Blockchain-based solution for the traceability of spare parts in manufacturing. IEEE Access, 8, 100308–100322. https://doi.org/10.1109/ACCESS.2020.2998159

Hastig, G. M., & Sodhi, M. S. (2020). Blockchain for supply chain traceability: Business requirements and critical success factors. Production and Operations Management, 29(4), 935–954.

Helo, P., & Ajmal, M. M. (2015). Conceptualizing trust in global context with focus on international projects and operations.

Henry, T., & Tucci-Piergiovanni, S. (2024). Secure proof verification blockchain patterns. International Conference on Business Process Management, 71–88.

Hevner, A. R., March, S. T., Park, J., & Ram, S. (2004). Design science in information systems research. MIS quarterly, 75–105.

Hoonsopon, D. (2022). Applications of blockchain in business processes: A comprehensive review.

Horita, H., Honda, K., Sei, Y., Nakagawa, H., Tahara, Y., & Ohsuga, A. (2014). Transformation approach from kaos goal models to bpmn models using refinement patterns. Proceedings of the 29th Annual ACM Symposium on Applied Computing, 1023–1024.

Hu, H., Ma, Q., Zhang, T., Tan, Y., Xiang, H., Fu, C., & Feng, Y. (2015). Semantic modelling and automated reasoning of non-functional requirement conflicts in the context of softgoal interdependencies. IET Software, 9(6), 145–156.

Hughes, L., Dwivedi, Y. K., Misra, S. K., Rana, N. P., Raghavan, V., & Akella, V. (2019). Blockchain research, practice and policy: Applications, benefits, limitations, emerging research themes and research agenda. International journal of information management, 49, 114–129.

Hussein, Z., Salama, M. A., & El-Rahman, S. A. (2023). Evolution of blockchain consensus algorithms: A review on the latest milestones of blockchain consensus algorithms. Cybersecurity, 6(1), 30. https://doi.org/10.1186/s42400-023-00163-y

Imeri, A., Agoulmine, N., Feltus, C., & Khadraoui, D. (2019). Blockchain: Analysis of the new technological components as opportunity to solve the trust issues in supply chain management. In K. Arai, R. Bhatia, & S. Kapoor (Eds.), Intelligent computing (pp. 474–493). Springer International Publishing.

Janssen, M., Brous, P., Estevez, E., Barbosa, L. S., & Janowski, T. (2020). Data governance: Organizing data for trustworthy artificial intelligence. Government information quarterly, 37(3), 101493.

Jayashri, N., Rampur, V., Gangodkar, D., M, A., C, B., & N, A. K. (2023). Improved block chain system for high secured iot integrated supply chain. Measurement: Sensors, 25, 100633. https://doi.org/https://doi.org/10.1016/j.measen.2022.100633

Jensen, T., Hedman, J., & Henningsson, S. (2019). How tradelens delivers business value with blockchain technology. MIS Quarterly Executive, 18(4).

Johng, H., Kim, D., Park, G., Hong, J.-E., Hill, T., & Chung, L. (2020). Enhancing business processes with trustworthiness using blockchain: A goal-oriented approach. Proceedings of the 35th Annual ACM Symposium on Applied Computing, 61–68.

Jovanovic, M., Kostić, N., Sebastian, I. M., & Sedej, T. (2022). Managing a blockchain-based platform ecosystem for industry-wide adoption: The case of tradelens. Technological Forecasting and Social Change, 184, 121981.

Kambilo, E. K., Rychkova, I., Herbaut, N., & Souveyet, C. (2023). Addressing trust issues in supply-chain management systems through blockchain software patterns. International Conference on Research Challenges in Information Science, 275–290.

Kambilo, E. K., Zghal, H. B., Guegan, C. G., Stankovski, V., Kochovski, P., & Vodislav, D. (2022). A blockchain-based framework for drug traceability: Chaindrugtrac. Proceedings of the 37th ACM/SIGAPP Symposium on Applied Computing, 1900–1907.

Kamble, S. S., Gunasekaran, A., & Sharma, R. (2020). Modeling the blockchain enabled traceability in agriculture supply chain. International Journal of Information Management, 52, 101967. https : / / doi .org/https://doi.org/10.1016/j.ijinfomgt.2019.05.023

Kastner, C., Thum, T., Saake, G., Feigenspan, J., Leich, T., Wielgorz, F., & Apel, S. (2009). Featureide: A tool framework for feature-oriented software development. 2009 ieee 31st international conference on software engineering, 611–614.

Khalifa, D., Madjid, N. A., & Svetinovic, D. (2019). Trust requirements in blockchain systems: A preliminary study. 2019 Sixth International Conference on Software Defined Systems (SDS), 310–313.

Kim, B. H. (2024). Vconmc: Enabling consistency verification for distributed systems using implementation-level model checkers and consistency oracles. Electronics, 13(6), 1153. https://doi.org/10.3390/electronics13061153

Kotusev, S. (2017). Enterprise architecture: What did we study? International Journal of Cooperative Information Systems, 26(04), 1730002.

Kuhn, M., Funk, F., Zhang, G., & Franke, J. (2021). Blockchain-based application for the traceability of complex assembly structures. Journal of Manufacturing Systems, 59, 617–630. https://doi.org/https://doi.org/10.1016/j.jmsy.2021.04.013

Lambert, D. M., & Cooper, M. C. (2000). Issues in supply chain management. Industrial marketing management, 29(1), 65–83.

Lauster, C., Klinger, P., Schwab, N., & Bodendorf, F. (2020). Literature review linking blockchain and business process management. Proc. 15th Int. Conf. Wirtschaftsinformatik, 1802–1817.

Le Berre, D., & Parrain, A. (2010). The sat4j library, release 2.2. Journal on Satisfiability, Boolean Modeling and Computation, 7(2-3), 59–64.

Lindell, Y. (2020). Secure multiparty computation. Communications of the ACM, 64(1), 86–96.

Liu, J., Shen, X., Xie, M., & Zhang, Q. (2022). Research on sharding strategy of blockchain based on topsis. International Conference on Smart Computing and Communication, 247–257.

López-Pintado, O., García-Bañuelos, L., Dumas, M., Weber, I., & Ponomarev, A. (2019). Caterpillar: A business process execution engine on the ethereum blockchain. Software: Practice and Experience, 49(7), 1162–1193.

Luhmann, N. (2017). Trust and power. Polity Press.

Mairiza, D., Zowghi, D., & Nurmuliani, N. (2010). An investigation into the notion of non-functional requirements. Proceedings of the 2010 ACM symposium on applied computing, 311–317.

Manning, L., Brewer, S., Craigon, P. J., Frey, J., Gutierrez, A., Jacobs, N., Kanza, S., Munday, S., Sacks, J., & Pearson, S. (2022). Artificial intelligence and ethics within the food sector: Developing a common language for technology adoption across the supply chain. Trends in Food Science & Technology, 125, 33–42. https://doi.org/https://doi.org/10.1016/j.tifs.2022.04.025

Marangunić, N., & Granić, A. (2015). Technology acceptance model: A literature review from 1986 to 2013. Universal Access in the Information Society, 14, 81–95. https://doi.org/10.1007/s10209-014-0348-1

Marques, L., & Pereira, R. (2023). Creating value with blockchain for organizations. In Digital technologies and transformation in business, industry and organizations (pp. 17–41). Springer.

Masudin, I., Rahmatullah, B. B., Agung, M. A., Dewanti, I. A., & Restuputri, D. P. (2022). Traceability system in halal procurement: A bibliometric review. Logistics, 6(4), 67.

Mattila, J., Seppälä, T., et al. (2016). Digital trust, platforms, and policy. ETLA Brief No, 42.

Mayer, R. C., Davis, J. H., & Schoorman, F. D. (1995). An integrative model of organizational trust. Academy of management review, 20(3), 709–734.

McGovern, J. (2004). A practical guide to enterprise architecture. Prentice Hall Professional.

Mcknight, J. B. T., M. Carter, & Clay, P. (2011). Trust in a specific technology: An Investigation of its Components and Measures. ACM Transactions on Management Information Systems, 2(12), 1–25.

Meeßen, S. M., Thielsch, M. T., & Hertel, G. (2020). Trust in Management Information Systems (MIS): A Theoretical Model. Zeitschrift fur Arbeits- und Organisationspsychologie, 64(1), 6–16.

Mellor, S. J. (2004). Mda distilled: Principles of model-driven architecture. Addison-Wesley Professional.

Mendling, J., Weber, I., Van Der Aalst, W., et al. (2018). Blockchains for business process management – challenges and opportunities. ACM Transactions on Management Information Systems (TMIS), 9(1), 4.

Mingxiao, D., Xiaofeng, M., Zhe, Z., Xiangwei, W., & Qijun, C. (2017). A review on consensus algorithm of blockchain. 2017 IEEE international conference on systems, man, and cybernetics (SMC), 2567–2572.

Mittelstadt, e. a. (2022). Ethics of algorithmic decision-making in the public sector: A systematic review. Government Information Quarterly, 39(1).

Morkunas, V. J., Paschen, J., & Boon, E. (2019). How blockchain technologies impact your business model. Business Horizons, 62(3), 295–306.

Morley, J. e. a. (2020). The debate on the ethical implications of artificial intelligence: A review and classification of arguments. Science and Engineering Ethics, 26, 1465–1483.

Mühlböck, M. e. a. (2022). System-based trust and platform governance in the digital economy. Information Systems Journal. https://doi.org/10.1111/isj.12354

Nakamoto, S. (2008). Bitcoin: A peer-to-peer electronic cash system un sistèma de pagament electronic.

Noel, R., Panach, I., Ruiz, M., & Pastor, O. (2021). The litestrat method: Towards strategic model-driven development. Proceedings of the 29th International Conference on Information Systems Development.

Papathanasiou, J., Ploskas, N., Papathanasiou, J., & Ploskas, N. (2018). Topsis. Springer.

Pastor, O., Noel, R., & Panach, I. (2021). From strategy to code: Achieving strategical alignment in software development projects through conceptual modelling, 145–164.

Patrick, A. S., Briggs, P., & Marsh, S. (2005). Designing systems that people will trust. Security and Usability, 1(1), 75–99.

Pietrzak, P., & Takala, J. (2021). Digital trust–asystematic literature review. Forum Scientiae Oeconomia 9(3), 59-71.

Pimentel, J., & Castro, J. (2018). Pistar tool–a pluggable online tool for goal modeling. 2018 IEEE 26th International Requirements Engineering Conference (RE), 498–499.

Pohl, K. (2010). Requirements engineering: Fundamentals, principles, and techniques. Springer Publishing Company, Incorporated.

Pournader, M., Kach, A., & Talluri, S. (2020). A review of the existing and emerging topics in the supply chain risk management literature. Decision sciences, 51(4), 867–919.

Rajasekar, V., Sondhi, S., Saad, S., & Mohammed, S. (2020). Emerging design patterns for blockchain applications. ICSOFT, 242–249.

Rajnak, V., & Puschmann, T. (2021). The impact of blockchain on business models in banking. Information Systems and e-Business Management, 19, 809–861.

Rajora, N. (2023). Pharmaceutical drug traceability by blockchain and iot in enterprise systems. Universal Journal of Pharmacy and Pharmacology, 11–18.

Ritter, J. (2020). Digital trust. https://www.techtarget.com/whatis/definition/digital-trust (accessed: 11.05.2022).

Ruiz, M., Costal, D., España, S., Franch, X., & Pastor, O. (2015). Gobis: An integrated framework to analyse the goal and business process perspectives in information systems. Information Systems, 53, 330–345. https://doi.org/10.1016/j.is.2015.03.007

Rusu, R., & Baboş, A. (2015). Organizational trust between institutional and interpersonal trust. Scientific Bulletin, 20(2), 55–60.

Rychkova, Irina. (2008). Formal semantics for refinement verification of entreprise models. https://doi.org/10.5075/EPFL-THESIS-4210

Seidl, D., & Mormann, H. (2014). Niklas luhmann as organization theorist. The Oxford handbook of sociology, social theory, and organization studies: Contemporary cu 125–157.

Six, N., Herbaut, N., & Salinesi, C. (2022a). Blockchain software patterns for the design of decentralized applications: A systematic literature review. Blockchain: Research and Applications, 3(2), 100061.

Six, N., Correa-Restrepo, C., Herbaut, N., & Salinesi, C. (2022). An ontology for software patterns: Application to blockchain-based software development. International Conference on Enterprise Design, Operations, and Computing, 284–299.

Six, N., Herbaut, N., & Salinesi, C. (2022b). Blockchain software patterns for the design of decentralized applications: A systematic literature review. Blockchain: Research and Applications, 100061.

Soundararajan, R., & Shenbagaraman, V. (2023). Unlocking the potential of blockchain through multi-criteria decision making in platform selection. International Journal of Professional Business Review: Int. J. Prof. Bus. Rev., 8(4), 49.

Spewak, S. H., & Hill, S. C. (1993). Enterprise architecture planning: Developing a blueprint for data, a QED Information Sciences, Inc.

Stafford, V. (2020). Zero trust architecture. NIST special publication, 800(207), 800–207.

Suárez-Figueroa, M. C., Gómez-Pérez, A., & Fernández-López, M. (2011). The neon methodology for ontology engineering. In Ontology engineering in a networked world (pp. 9–34). Springer.

Sumesh, S., & Krishna, A. (2022). Challenges and review of goal-oriented requirements engineering based competitive non-functional requirements analysis. Multiagent and Grid Systems, 18(2), 171–191.

Taddeo, M. (2009). Defining trust and e-trust: From old theories to new problems. International journal of technology and human interaction (IJTHI), 5(2), 23–35.

Tran, A. B., Lu, Q., & Weber, I. (2018). Lorikeet: A model-driven engineering tool for blockchain-based business process execution and asset management. BPM (dissertation/demos/industry), 56–60.

Trinidad, P., Cortés, A. R., Pena, J., & Benavides, D. (2007). Mapping feature models onto component models to build dynamic software product lines. SPLC (2), 51–56.

Van Lamsweerde, A. (2001). Goal-oriented requirements engineering: A guided tour. Proceedings fifth ieee international symposium on requirements engineering, 249–262.

Vijayaraj, N., Rajalakshmi, D., Immaculate, P., Sathianarayani, B., Rajeswari, S., & Gomathi, S. (2024). An innovative approach to improve the quality of pharmaceuticals approach using cloud computing. EAI Endorsed Transactions on Pervasive Health and Technology, 10.

Viriyasitavat, W., & Martin, A. (2011). A survey of trust in workflows and relevant contexts. IEEE Communications Surveys & Tutorials, 14(3), 911–940.

Vlachogianni, P., & Tselios, N. (2022). Perceived usability evaluation of educational technology using the System Usability Scale (SUS): A systematic review. Journal of Research on Technology in Education, 54(3), 392–409. https://doi.org/10.1080/15391523.2021.1929032

Voloch, N., Nissim, P., Elmakies, M., & Gudes, E. (2019). A role and trust access control model for preserving privacy and image anonymization in social networks. Trust Management XIII: 13th IFIP WG 11.11 International Conference, IFIPTM 2019, Copenha 19–27.

Wang, J., Yan, Z., Wang, H., Li, T., & Pedrycz, W. (2022). A survey on trust models in heterogeneous networks. IEEE Communications Surveys & Tutorials, 24(4), 2127–2162.

Wang, Y. D., & Emurian, H. H. (2005). An overview of online trust: Concepts, elements, and implications. Computers in human behavior, 21(1), 105–125.

Wegmann, A., & Preiss, O. (2003). Mda in enterprise architecture? the living system theory to the rescue. Seventh IEEE International Enterprise Distributed Object Computing Conference, 2003. Proc 2–13.

Wegmann, A., Regev, G., Rychkova, I., Lê, L.-S., De La Cruz, J. D., & Julia, P. (2007). Business and it alignment with seam for enterprise architecture. 11th IEEE international enterprise distributed object computing conference (EDOC 2007), 111–111.

Wei, Y. (2020). Blockchain-based data traceability platform architecture for supply chain management. 2020 IEEE 6th Intl Conference on Big Data Security on Cloud (BigDataSecurity) 77–85. https://doi.org/10.1109/BigDataSecurity-HPSC-IDS49724.2020.00025

Wieringa, R. (2014). Design science methodology for information systems and software engineering. Springer.

Xu, X., & Choi, T.-M. (2021). Supply chain operations with online platforms under the cap-and-trade regulation: Impacts of using blockchain technology. Transportation Research Part E: Logistics and Transportation Review, 155, 102491.

Xu, X., Pautasso, C., Lo, S. K., Zhu, L., Lu, Q., & Weber, I. (2025). An extended pattern collection for blockchain-based applications. In Transactions on pattern languages of programming v (pp. 67–117). Springer.

Xu, X., Pautasso, C., Zhu, L., Lu, Q., & Weber, I. (2018). A pattern collection for blockchain-based applications. Proceedings of the 23rd European Conference on Pattern Languages of Programs, 1–20.

Yan, Z., & Holtmanns, S. (2008). Trust modeling and management: From social trust to digital trust. Computer security, privacy and politics: current issues, challenges and solutions 290–323.

Yu, Y., Lapouchnian, A., Liaskos, S., Mylopoulos, J., & Leite, J. C. (2008). From goals to high-variability software design. Foundations of Intelligent Systems: 17th International Symposium, ISMIS 2008 Toronto, Can 1–16.

Zheng, Z., Xie, S., Dai, H.-N., Chen, X., & Wang, H. (2018). Blockchain challenges and opportunities: A survey. International journal of web and grid services, 14(4), 352–375.