Trust-aware Business Processes with Distributed Ledger Technologies

vorgelegt von
M. Sc.
Marcel Müller
ORCID: 0000-0001-9767-6426

an der Fakultät IV – Elektrotechnik und Informatik
der Technischen Universität Berlin
zur Erlangung des akademischen Grades

Doktor der Ingenieurwissenschaften
- Dr.-Ing. -

genehmigte Dissertation

Promotionsausschuss:

Vorsitzender: Prof. Dr. Jens Lambrecht
Gutachterin: Prof. Dr. Axel Küpper
Gutachter: Prof. Dr. Ingo Weber
Gutachter: Prof. Dr. Michael Rosemann
Gutachter: Prof. Dr. Peter Ruppel

Tag der wissenschaftlichen Aussprache: 13. Oktober 2021

Berlin 2021

Abstract

The progressing digital transformation and internationalization of business processes cause a shift towards a more collaborative nature of processes. In such collaborations, different organizations execute separate parts of the process autonomously. This fragmentation leads to uncertainty regarding the correct execution of activities, the proper workflow, and data flow. If organizations engage in business together, trust is required. Current research identified blockchain and distributed ledger technologies as promising tools to mitigate trust issues in business processes. Yet, the detailed relationship of distributed ledgers and trust is regarded ambivalently.

This thesis proposes three design science artifacts for analyzing and creating trust-aware business processes using distributed ledger technologies. Firstly, *TAPE* is a method for trust-aware business process engineering. The TAPE method consists of three steps. The first step analyzes trust issues in business processes. Therefore, TAPE uses a business process model and a trust model for joint semantical analysis. TAPE's second step uses trust patterns to mitigate trust issues in a collaborative business process. The last step of the TAPE method implements the trust-aware process. The second artifact that this thesis proposes is *Trust Studio*. Trust Studio is a web application that aids process analysts and engineers in applying the TAPE method. The last artifact this thesis introduces is a set of *distributed ledger trust patterns*. These trust patterns are classified with a taxonomy that helps to identify their trust-enhancing capabilities.

A set of different evaluation methods assesses the utility of the introduced design science artifacts. The evaluation shows that all three contributions reached the status of a *minimal viable artifact*. Thus, the outcomes of this thesis foster a deep understanding of trust in business processes and how distributed ledger technologies can be utilized as a tool to mitigate trust issues.

Zusammenfassung

Die fortschreitende digitale Transformation und die Internationalisierung von Geschäftsprozessen führen zu einer Verlagerung hin zu einer stärkeren Kollaboration von Prozessen. Bei solchen Kollaborationen führen verschiedene Organisationen separate Teile des Prozesses autonom aus. Diese Fragmentierung führt zu Unsicherheiten hinsichtlich der korrekten Ausführung von Aktivitäten, des richtigen Workflows und des Datenflusses. Wenn Organisationen gemeinsam Geschäfte machen, ist Vertrauen erforderlich. Aktuelle Forschungen haben Blockchain und Distributed Ledger Technologien als vielversprechende Ansätze identifiziert, um Vertrauensprobleme in Geschäftsprozessen zu entschärfen. Die genaue Beziehung von verteilten Ledgern und Vertrauen wird jedoch ambivalent betrachtet.

In dieser Arbeit werden drei Design-Science-Artefakte für die Analyse und Erstellung vertrauenswürdiger Geschäftsprozesse unter Verwendung von Distributed Ledgers vorgeschlagen. Erstens, TAPE ist eine Methode für vertrauenswürdiges Business Process Engineering. Die TAPE-Methode besteht aus drei Schritten. Im ersten Schritt werden Vertrauensprobleme (trust issues) in Geschäftsprozessen analysiert. Dazu verwendet TAPE ein Geschäftsprozessmodell und ein Vertrauensmodell zur gemeinsamen semantischen Analyse. Der zweite Schritt von TAPE verwendet Vertrauensmuster (trust patterns), um Vertrauensprobleme in einem kollaborativen Geschäftsprozess zu entschärfen. Der letzte Schritt der TAPE-Methode implementiert den vertrauens-bewussten Prozess. Das zweite Artefakt, das diese Arbeit vorschlägt, ist Trust Studio. Trust Studio ist eine Webanwendung, die Prozessanalysten und Ingenieure bei der Anwendung der TAPE-Methode unterstützt. Das letzte Artefakt, das diese Arbeit vorstellt, ist eine Sammlung von Distributed Ledger Vertrauensmuster (DLT trust patterns). Diese Vertrauensmuster werden anhand einer Taxonomie klassifiziert, die hilft, ihre vertrauensfördernden Fähigkeiten zu identifizieren.

Eine Reihe von verschiedenen Evaluationsmethoden bewertet den Nutzen der eingeführten Design-Science-Artefakte. Die Evaluierung zeigt, dass alle drei Beiträge den Status eines minimal brauchbaren Artefakts erreicht haben (minimal viable artifact). Somit fördern die Ergebnisse ein tiefes Verständnis von Vertrauen in Geschäftsprozessen und wie Distributed-Ledger-Technologien als Werkzeug zur Abschwächung von Vertrauensproblemen genutzt werden können.

Acknowledgements

I would like to thank all the people that supported me, gave me advice, and collaborated with me throughout the work on this thesis.

First of all, I would like to thank Prof. Dr. Axel Küpper for his support and valuable guidance over the years. The fruitful discussions with him and his support for my ideas were inalienable for me to complete this work. Furthermore, I would like to thank Prof. Dr. Peter Ruppel, Prof. Dr. Michael Rosemann, and Prof. Dr. Ingo Weber for their feedback. Validating my ideas with people who are even more passionate about their work than I am supported and motivated me immensely.

I would like to thank the students who helped to work on topics related to this thesis. Especially, this thank goes to Jacek Janczura, Jonathan Rau, Denis Koljada, and Thu-My Huynh for their tireless efforts. Furthermore, I would like to thank my colleagues within and outside of SNET for all the good discussions and idea exchanges we had over the years. This thank goes especially to Sandro Rodriguez Garzon and Nadine Ostern. They have been a reliable source of feedback and support throughout the years.

I would like to thank the European Commission and EIT Digital for funding the projects that helped me shape the contribution of my thesis.

Finally, I would like the thank all my friends and my family for standing by me at any time. Without their support and patience with me, I would not have finished this thesis.

June 2021, Berlin, Germany

Marcel Müller

Contents

1 Introduction

In recent years, many business processes have become more complex and collaborative in their nature. Two significant trends illustrate that shift. As a first trend, the progressing *globalization of business processes* fragments them between distinct business partners. For example, in global supply chains, many separate organizations have to collaborate to create a product and deliver it to the consumer [1, 2]. This process includes manufacturers and different logistic companies that transport parcels worldwide and interact with each other. As a second trend, with the *emerging digital economy* many multi-sided platforms evolved that allow users to engage in business with partners they have never interacted with before [3]. Such platforms let users rent apartments from unknown landlords (AirBnB)[1], share their cars with strangers (BlaBlaCar)[2], or buy goods from previously unknown merchants (eBay)[3].

These two trends have in common that the underlying business processes transformed into a more collaborative nature with more involved actors. It is characteristic of such collaborative business processes that subprocesses executed by autonomous collaborators are independent of each other. This causes uncertainty regarding the desired execution of the subprocess for the other collaborators. The buyer of a product in an e-commerce store might wonder whether the products will reach her in time for holiday season. The user of a ride-sharing platform might wonder if the stranger's car will bring her safely to her destination. When there is uncertainty between different parties in a process, there is a need for trust.

Blockchain and distributed ledger technologies (DLTs) promise to be a solution to trust issues in collaborative business processes [4]. Yet, current literature lacks a formal definition of how DLTs achieve trust. Furthermore, trust is a highly social concept. This makes it hard to analyze trust issues and mitigate them in an engineering fashion.

This thesis focuses on the research question "can trust issues in business collaborations be systematically analyzed and mitigated by using technological innovations, such as distributed ledger technologies?". Therefore, this thesis proposes *TAPE*, a novel methodological framework to analyze and mitigate trust issues in collaborative business processes. TAPE uses *TRUB*, a new trust model and *Trust Mining* as an automated method to analyze trust issues. Further, this thesis defines trust patterns that can mitigate trust issues. Therefore, the focus of this thesis is centered on how blockchain and DLTs can reduce trust issues in collaborative business processes and how to implement such applications.

[1] https://www.airbnb.com
[2] https://www.blablacar.de
[3] https://www.ebay.com

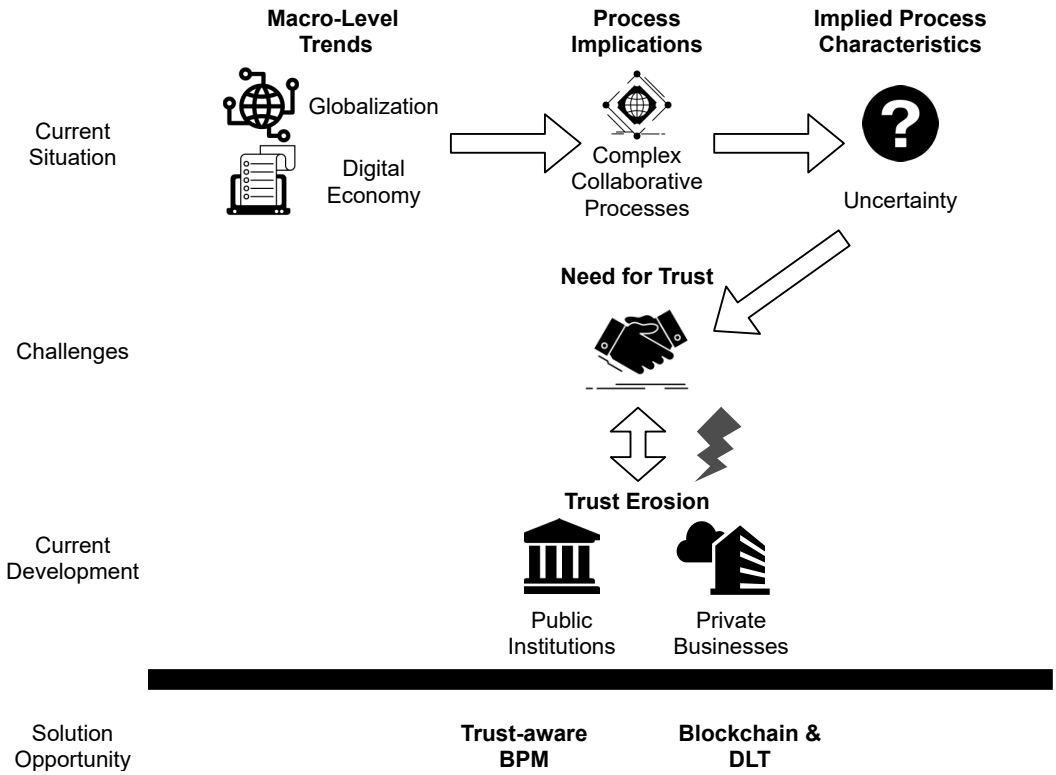

Figure 1.1: Motivational Outline

1.1 Motivation

Recently, many business processes experienced a change in their core characteristics. More complex processes evolved with more participating parties. The advancing globalization and the evolving digital economy are two main drivers of the macro-level trends. The following sections elaborate on these trends and their implications on trust in business processes. Studies on trust erosion regarding these changes illustrate the obstacles to overcome to enable further development in complex collaborative business processes.

1.1.1 Macro-level Trends in Business Processes

Global value chains (GVCs) have expanded their share in overall world-wide production greatly over the past three decades [5]. In such chains, different companies specialize in executing a distinct set of activities needed in a value-adding process. Many different organizations located in different countries work together collaboratively. In that way, the production process of goods is spread across the whole world. The progressing increase of importance of GVCs is especially visible in industries with large research and development needs before the mass production of products [5]. This often leads to complex GVCs with complex collaborative business processes. Examples for such complex processes are the production of high-tech products such as computers, the supply chain orchestrating the transportation of the components, and the final product. Further, especially e-commerce processes that aim to sell the final products to consumers have experienced further internationalization as well [6].

The trend to more collaborative and complex value-adding processes is not only limited to the creation of physical goods. The advance of the digital economy implies a similar shift in business processes. The term digital economy describes the economic activity "that results from billions of everyday online connections among people, businesses, devices, data, and processes" [7]. It is backed by the Internet, enabling connectivity between different people, machines, and organizations. The digital economy enabled new digital business processes. For example, the Internet of Things (IoT) harnesses the power of the provided interconnectivity for machine communication. IoT is defined as "a global infrastructure for the information society, enabling advanced services by interconnecting (physical and virtual) things based on existing and evolving interoperable information and communication technologies" [8]. Machine-to-machine interaction and autonomous machines are two examples of IoT-enabled use cases and illustrate one shape of the emerging Digital Economy.

A different shape of the Digital Economy is the evolution of multi-sided platforms that enable unknown partners to engage in business. Multi-sided platforms for apartment rental (e.g. AirBnB[4]) let tourists rent vacation homes from locals. Also, the last years have seen an explosion of mobility sharing services that let users share rides (e.g., BlaBlaCar[5]), or lease vehicles (e.g., e-scooter sharing such as Lime [6]). The business processes in this Sharing Economy [3] are also involving different collaborators and are increasingly complex by their nature.

The collaborative and complex value-adding processes of these macro-level trends have common characteristics. One of the most significant is the independence of different collaborators. Activities in subprocesses that one organization is responsible for are usually out of the realm of influence of all other organizations collaborating in the process. This isolation of influence in certain parts of the business processes leads to process *uncertainty* for the other collaborators and external process stakeholders. For example, in a global supply chain with different involved logistics organizations, many questions arise that illustrate uncertainties. The questions on whether the carrier arrives at the right time to hand over a parcel to the next carrier or whether all carriers handle the parcel with care are some examples of uncertainties relevant to process collaborators and stakeholders. When uncertainty exists in a business process, trust is required [9]. Trust can be seen as the subjective assessment of the uncertainties and the decision of engaging in the process despite the presence of uncertainty [10]. Hence, due to their distributed characteristics, collaborative business processes are trust-intensive by their nature.

1.1.2 Trust Erosion

While macro-level trends to a more collaborative nature of business processes require trust, at the same time trust erosion evolved to a major problem in the 21st century. The Edelman Trust Barometer [11] indicates that, despite a good

[4] https://www.airbnb.de
[5] https://www.blablacar.de
[6] https://www.li.me/

global economic situation[7] trust has become a major issue of modern society. The study showed that none of four observed societal institutions (government, business, NGOs and media) is trusted by a majority of the population in many countries. Concerns about the future of employment, the consumption of fake news and the distrust of societal leaders are major manifestations of the general distrust of societal organizations. Business processes are a central artifact of institutions. Hence, the generalized trust erosion also implies a distrust regrading the processes of them.

The Edelman Trust Barometer mainly focuses on business-to-consumer (B2C) interactions. Yet, the past years have shown a simplification of access to traditional business interactions for natural persons. Digital platforms like Shopify [8] enable micro-entrepreneurs to market and sell products to consumers. Thus, these small businesses can act like traditional companies. Yet, when interacting with large cooperations, they are still in a position as a customer. Hence, the Edelman study has implications for B2C and B2B (especial micro-entrepreneurs) interactions alike.

Potential origins of this experienced distrust in institutions and their business processes can be found in an plethora of incidents where existing trust towards different institutions has been harmed. The Cambridge Analytica Scandal [12] shows how serious violations of personal data of Facebook's users might have lead to an unfair advantage in the 2016 U.S. elections. From a process perspective, Facebook's user had the firm belief that the company is not misusing personal data in their advertising selling process. The German automotive sector has been for years one of the major drivers of the country's economy. The Volkswagen Scandal [13] revealed that the company manipulated their cars systematically to pass emission test. Customers and business partners alike both trusted in the processes executed by Volkswagen. When the manipulations gained public attention increase distrust towards the company could be observed [14].

The list of trust-harming incidents is non-exhaustive and can be continued with many other scandals. It illustrates how important trust is in business processes. Hence, analyzing trust concerning business processes as experienced an increasing academic and professional interest. There is a need to model and analyze trust in business processes in a formal way. With that as a base, trust-aware business processes can be constructed.

1.1.3 Trust-enhancement with Distributed Ledger Technologies

Distributed ledger technologies (DLTs) are decentralized approaches to store and modify state information of particular pieces of data in a tamper-proof way. This happens by using transactions to change state information. Transaction submitters propose their transactions to a decentralized network. The peers maintain a shared state of the ledger. Therefore, they validate new transactions and engage in a consensus protocol. A blockchain is a special DLT that orders new transactions in a collection called blocks. Peers in a blockchain network validate these transactions periodically. Validated transactions are ordered in new blocks

[7] 2020 before the COVID-19 mass outbreak
[8] https://www.shopify.com

that are cryptographically linked to all previous blocks, creating the metaphorical chain. Through the employed cryptography, older transactions are practically tamper-resistant, as long as not a majority of the network peers decides to act maliciously. This makes blockchain and DLTs a good candidate to improve trust issues in processes.

In its initial implementations, the blockchain technology was centered around applications in the finance domain [15]. Recently, other application domains evolved. In the finance domain, transactions are used to transfer cryptographic assets between different owners. During further development, newer generations of blockchain technologies evolved that enabled users to state information of arbitrary objects through the utilization of smart contracts [16]. These enabled more general applications in other domains such as supply chain management, healthcare, and the energy sector [17]. Further, DLTs have also been described as a technology which is especially useful for novel applications in the progressing Digital Economy [18].

The advancing technical capabilities of blockchain and DLTs inflicted an increasing professional and academic interest in using such technologies for business process management (BPM) in general [4]. Prior to this thesis, research on BPM and distributed ledgers mostly centered on implementation aspects. A deeper investigation on how blockchain and DLTs exactly improve trust issues in collaborative business processes has not been carried out.

1.2 Methodology

The methodological approach of this work is illustrated in Figure 1.2. It proposes to have a basic research question that can be translated into hypotheses. The hypotheses can be formulated as problems that may be solved using the design science paradigm [19] in information systems research. Solution approaches aim to solve the fundamental problems. Predictions claim the utility of the solution approaches, which are verified by different evaluations.

The overall research question is defined as follows:

- Q1 How can trust issues in business collaborations be systematically analyzed in an automated fashion and mitigated by using technological innovations, specifically distributed ledger technologies?

This question reflects the need for research on the feasibility of a systematic approach towards trust in business collaborations on the one side and on applying it in an engineering fashion by using distributed ledger technologies on the other hand. Therefore, two hypotheses are formulated regarding the research question.

- H1 Improving trust issues in business collaborations can be supported by a process-centered design method.
- H2 Distributed ledger technologies can be used in different ways to mitigate trust issues in business collaborations.

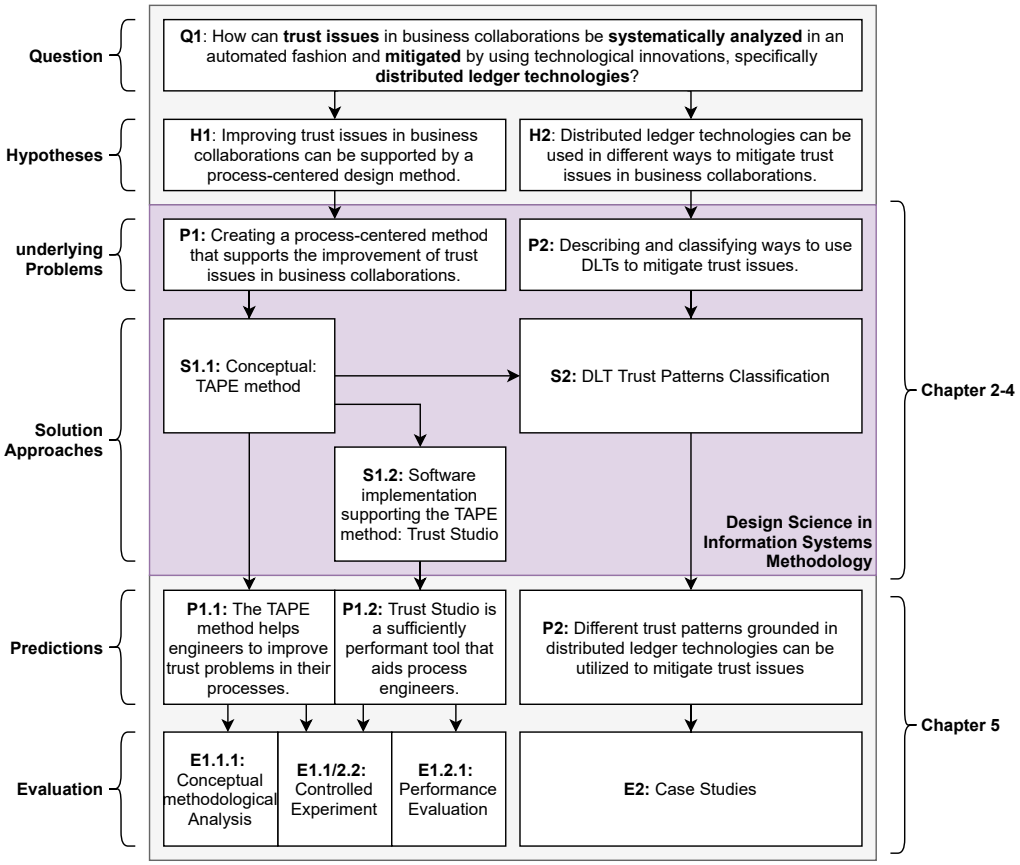

Figure 1.2: Methodological outline of this thesis.

These hypotheses are phrased so that it is possible to create systematic methods for mitigating trust issues in a process-centered fashion, implying an exploratory research methodology. Therefore, this work proposes to follow the design science [19] research paradigm. Design science in information systems research focuses on *IT artifacts*, such as constructs (vocabulary and symbols), models (abstractions and representations), methods (algorithms and practices), and instantiations (implemented in prototype systems) [19]. These artifacts are created to solve certain problems. Hypotheses H1 and H2 can be rephrased as problems that reflect the challenges to address as an input to a design science research method.

- P1 Creating a process-centered method that supports the improvement of trust issues in collaborative business processes.
- P2 Describing and classifying ways to use DLTs to mitigate trust issues.

Creating a process-centered method that supports the improvement of trust in business collaborations is the underlying problem that needs to be solved so that hypothesis H1 can be supported. Equivalently, describing and classifying ways to use distributed ledgers analytically and constructively to mitigate trust issues in business collaborations is the underlying problem of hypothesis H2. The following three solution approaches aim to solve these two problems:

- S1.1 Conceptual method: TAPE

- S1.2 Implementation of TAPE: Trust Studio
- S2 Distributed ledger trust patterns classification

Solution approaches S1.1 and S1.2 aim to support hypothesis H1 by solving problem P1. Solution approach S2 targets hypothesis H2 by solving problem P2. The first conceptual contribution of this work towards improving trust in business collaborations uses business process management as a methodology from the domain model-driven software engineering and proposes a conceptual method for trust-aware business process engineering (TAPE). As a tool to support the conceptual method, S1.2 implements Trust Studio. The software tool enables to model and analyze trust in collaborative business processes. The TAPE method consists of three steps: analyzing trust in collaborative business processes (I), mitigating trust issues with trust-enhancing tools and technologies (II) and implementing the trust-enhanced process (III). Hypothesis H2 asserts that there is a way to systematically use distributed ledger technologies in different manners to mitigate trust issues business collaborations systematically. Therefore, S2 uses the TAPE method and investigates deeper the role of DLTs in steps II and III. This approach proposes a trust-centered classification of different design patterns using DLTs. In the terms of the design science research paradigm the TAPE method, Trust Studio and the DLT trust patterns are IT artifacts.

For each of these three solution approaches, predictions assert how far they can provide utility to solve the problems and support the hypotheses.

- P1.1 The TAPE methods helps engineers to improve trust problems in their collaborative business processes.
- P1.2 Trust Studio is a sufficiently performing and usable tool that aids process engineers applying the TAPE method.
- P2 Different trust patterns rooted in distributed ledger technologies can be utilized to mitigate trust issues.

Prediction P1.1 claims that the TAPE is useful and valuable for process engineers. P1.2 claims that Trust Studio is a useful tool that supports the application of the TAPE method. P2 claims that the application of the trust patterns defined in S2 improves trust issues that could hardly be addressed by non-DLT approaches. To verify the predictions, four different evaluation approaches for the scientific contribution of this work are discussed.

- E1.1.1 Conceptual-methodological Analysis of TAPE
- E1.1.2/2.2 Controlled Experiment
- E1.2.1 Performance Test
- E2 Case Study

The prediction that TAPE helps engineers to improve their trust problems P1.1 is evaluated and analyzed on an argumentative level using a conceptional--methodological analysis (E1.1.1) and through controlled experiments with experts (E1.1.2/2.2). The controlled experiments uses TAPE and the DLT patterns

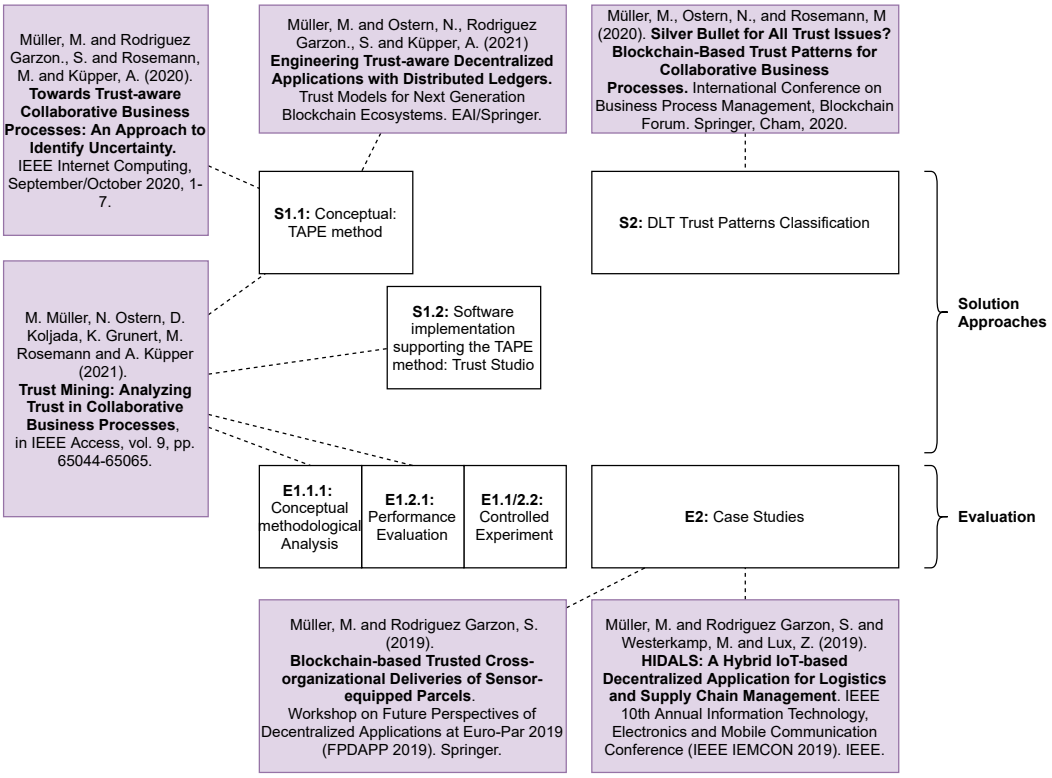

Figure 1.3: Overview of pre-publications of this thesis.

in a guided tutorial with focus on usability and ease of use. These characteristics are firstly concerned with the software implementation Trust Studio. Since Trust Studio implements the TAPE method, the controlled experiment enables insights to both the method and its implementation. Additionally, Trust Studio is evaluated separately regarding its performance (E1.2.2). The trust-enhancing capabilities of the distributed ledger trust patterns are evaluated in case studies (E2).

1.3 Publications

Parts of this thesis have been published in the following publications:

Book Chapters

- Müller, M. and Ostern, N., Rodriguez Garzon., S. and Küpper, A. (2021) Engineering Trust-aware Decentralized Applications with Distributed Ledgers. Trust Models for Next Generation Blockchain Ecosystems. EAI/Springer.

Journal Papers

- M. Müller, N. Ostern, D. Koljada, K. Grunert, M. Rosemann and A. Küpper (2021). Trust Mining: Analyzing Trust in Collaborative Business Processes, in IEEE Access, vol. 9, pp. 65044-65065.

Magazine Publications

- Müller, M. and Rodriguez Garzon., S. and Rosemann, M. and Küpper, A. (2020). Towards Trust-aware Collaborative Business Processes: An Approach to Identify Uncertainty. IEEE Internet Computing, September/October 2020, 1-7.

Conference Papers

- Müller, M., Ostern, N., and Rosemann, M (2020). Silver Bullet for All Trust Issues? Blockchain-Based Trust Patterns for Collaborative Business Processes. International Conference on Business Process Management, Blockchain Forum. Springer, Cham, 2020.
- Müller, M. and Rodriguez Garzon, S. and Westerkamp, M. and Lux, Z. (2019). HIDALS: A Hybrid IoT-based Decentralized Application for Logistics and Supply Chain Management. 2019 IEEE 10th Annual Information Technology, Electronics and Mobile Communication Conference (IEEE IEMCON 2019). IEEE.

Workshop Papers

- Müller, M. and Rodriguez Garzon, S. (2019). Blockchain-based Trusted Cross-organizational Deliveries of Sensor-equipped Parcels. Workshop on Future Perspectives of Decentralized Applications at Euro-Par 2019 (FP-DAPP 2019). Springer.

The following chapters are marked regarding their respective pre-publication of contents. Figure 1.3 schematically illustrates the connection of the scientific contributions to the publications.

1.4 Outline

The remainder of this thesis is structured as follows. Chapter 2 highlights related work on the three conceptual pillars of this work: trust, business process management, and distributed ledgers. This section identifies a research gap at the intersection of the three topics. Chapter 3 establishes the scientific contributions. This includes the TAPE method and a classification of trust patterns with distributed ledgers that can be used with the TAPE method. Afterward, Chapter 4 discusses Trust Studio. This software tool aims to support the application of TAPE and the trust patterns to mitigate trust issues in inter-organizational processes. Chapter 5 evaluates the proposed concepts through a variety of different methods. Finally, Chapter 6 summarizes the outcome of this work and highlights future work.

2 Background and Related Work

This chapter discusses the background and related work of trust-aware business process design and distributed ledger technologies (DLTs).

Conceptionally, the foundations of this work are rooted in computer science, business management, and social science, as illustrated in Figure 2.1. The scientific contributions build upon three main pillars. Trust (1) as an inter-personal concept is the central subject of this work. Business process management (2) is the model-driven vehicle to express inter-organizational workflows and conceptualize trust therein. Distributed ledger technologies (3) are the tool of choice to mitigate trust issues within this thesis.

Section 2.1 elaborates on the concept of trust on a general level. It compares the concept of trust across various domains. Therefore, different conceptual definitions of trust and related concepts serve as a base for the contributions of this thesis. After the conceptual view on trust, Section 2.2 provides an overview of the research discipline of business process management (BPM). Afterward, the technical foundations of blockchain and DLTs are discussed in Section 2.3. Section 2.4 reviews current work that focuses on blockchain and DLTs for BPM.

2.1 Foundations of Trust

Trust is an abstract social concept. It is perceived differently in parts of society and business relationships. In particular research fields, its definition varies strongly. This section summarizes how specific research areas define trust in specific contexts. The concept is inspired by surveys by Grandison and Sloman [20] or Viriyasitavat and Martin [21].

Gambetta established one of the classic definitions of trust that is often used in sociology and other related research fields [22].

> "When we say we trust someone or that someone is trustworthy, we implicitly mean that the probability that he 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. Correspondingly, when we say that someone is untrustworthy, we imply that the probability is low enough for us to refrain from doing so." [22]

This definition implies a subjective assessment when a person decides to trust somebody or something. Furthermore, this definition reflects *uncertainty* being present in a situation where trust is needed. If there was no possibility that

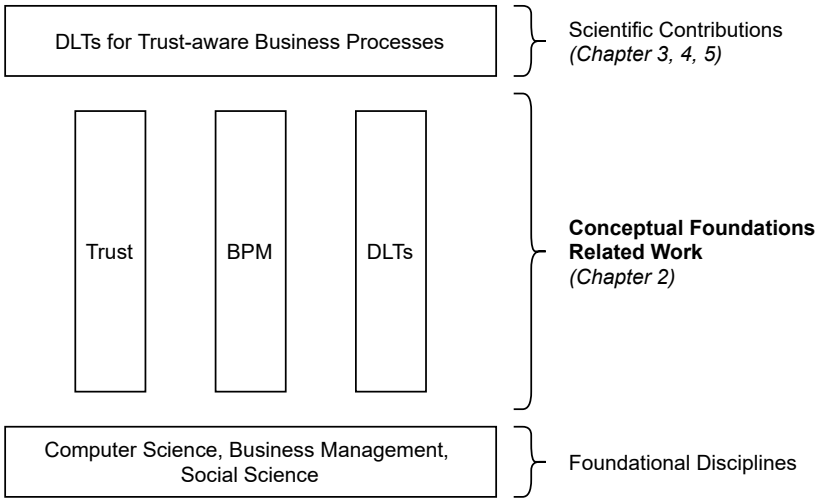

Figure 2.1: The foundations of this work are rooted in trust, business process management, and distributed ledger technologies.

anything undesired could happen, there would be no need for trust. Mayer et al. [23] established a broadly-used model for trust in an organizational context. They define the concept of trust in general as:

> "The willingness of a party to be vulnerable to the actions of another party based on the expectation that other will perform a particular action important to the trustor, irrespective of the ability to monitor or control that other party." [23]

This definition illustrates the situation when trust is needed and the impacts it can have. Trust is required when a one party (the *trustor*) has no control over another party's (the *trustee*) executed actions. This emphasizes uncertainty again and makes trust an essential subject for inter-organizational business processes. It is characteristic of such processes that activities carried out by one organization are outside the domain of control of all other organizations. Mayer et al. describe the impact of something that happens contrary to the expectation as *vulnerability* [23]. Lau and Whyte [24] define this concept in the context of automated organizational collaborations similar, exposing oneself to potential hazardous actions of others:

> "An act of trust involves placing yourself at hazard to another's actions, in a belief, at least partly without explicit computability of risk that they will act to your benefit." [24]

Furthermore, the phrasing of trust as an act in this definition also shows that trust is an active decision. Grandison and Slowman [20] comprised a survey on trust in Internet applications where they define trust as follows:

> "Trust is the firm belief in the competence of an entity to act dependably, securely, and reliably within a specified context." [20]

This definition reflects the "firm belief" in the competence of an entity as a positive sentiment. It is in contrast to the negative phrasing of exposing

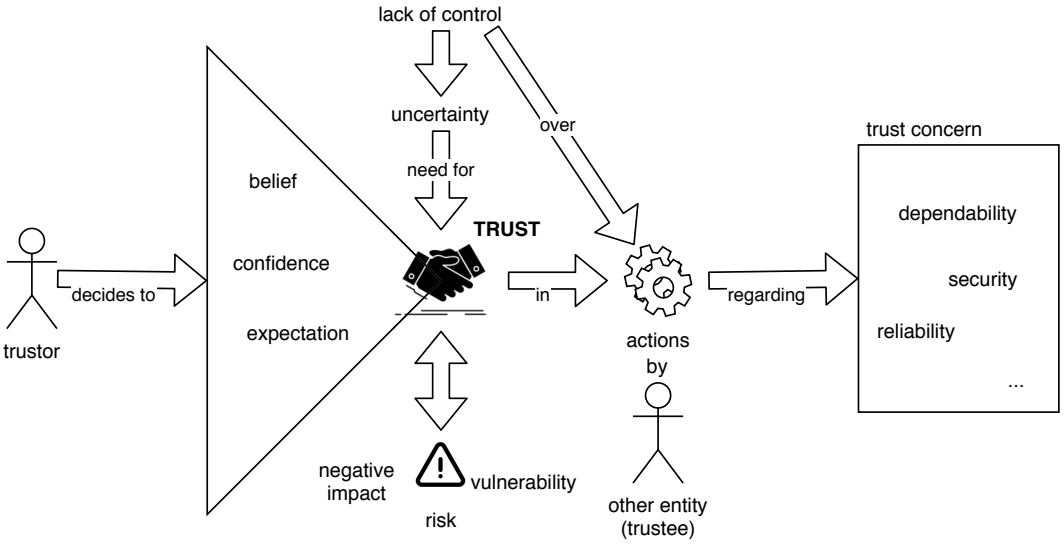

Figure 2.2: Sketch of the concept of trust with related concept and forces interaction with each other. Block arrows indicate interactions, not causality per se.

oneself to others' potential hazardous actions, as stated by Lau and Whyte [24]. This definition reflects that there may be different subjects of trust. Grandison and Sloman enumerated dependability, security, and reliability as important trust subjects. These subjects are highly domain-specific and may vary. These subjects are often referred to as *trust concerns*. There is no consensus on trust concerns established in current literature.

Since there is no consensus on the concept of trust, this section illustrates informally the forces and concepts that are related to trust, based on the previously discussed definitions, as shown in Figure 2.2. The lack of control over actions by others causes uncertainty. Uncertainty implies the need for trust. Trust in the actions of others has different trust concerns. The concept of trust itself can be described as the belief, the confidence, or the positive expectation that actions are executed positively for oneself even though there is a risk of negative impacts. This manifestation of negative impacts can be regarded as vulnerability. Trust is the acceptance of the risks with possible outcomes by a trustor that depends on a trustee.

2.1.1 The Role of Trust in Business Relationships

"Without trust, business as we know it, is not possible" [26]. This statement by Audi illustrates how trust is a necessity for the viability of any business. Thus, it is important for businesses to understand the major concepts related to trust. Castaldo et al. conducted a quantitative study on the concepts related to trust in business relationships [25]. In their analysis, they collected definitions of trust from scientific publications over a 20-year period. They extracted the most frequently used terms in the definitions of trust in the analyzed papers. Figure 2.3 illustrates the most frequently occurring terms and concepts. In their study, they grouped terms that are used for the same subjects. For instance, the

Figure 2.3: Illustration of the trust-related concepts found in [25]. Data extracted from Table 1 [25], own illustration. The size of the words corresponds to the frequency of occurrence in the definitions.

term actor subsumes the terms agent, company, group individual, and others. These terms refer semantically to the same concept.

The empirical study shows that the most important component related to trust is the trustee (the one in which trust is placed). Trust in this actor is often related to the actions (behaviors, acts) this actor executes. More specifically, the positive expected outcome or consequence of the actions are the focus of the trust definitions. Positive expectations can also be phrased as a belief or the will to rely on a trustee for certain actions. Such a trust relationship is often tightly coupled with risk.

2.1.2 Trust in a Globalized Business Interactions

The ongoing digital transformation and globalization of business interactions in business-to-business relationships also require trust. In a competitive world, joint ventures and non-equity collaborative agreements among global firms are trust-intensive. Thus, research on a better understanding of these arrangements and on how to increase their effectiveness has been conducted in the past [27]. Therefore, the relationship between contractual agreements and trust has been studied extensively. In this research field, it is commonly agreed that formal contracts and relational governance support business collaborations, while trust still has a critical role in business-to-business relationships [28, 29, 30].

2.1.3 Trust in the Digital Age

With the ongoing digital transformation and internationalization of business interactions, the role of trust gradually shifted. Castelfranchi and Falcone state that that trust and control are linked together [31]. This link has a strong influence on businesses and their change in the digital age. Traditional business

relationships are built on a history of previous collaboration and control thereof. With the advent of e-commerce in the early 2000s, this paradigm changed. Consumers could directly purchase goods from vendors they have not known before. Thus, there is no history of previous collaboration or means to control the business partner's actions in such a situation.

In this age, *trust management systems* emerged [32]. Trust management systems are mostly concerned with previous experiences and the reputation of certain actors in e-commerce. Reputation is the perception a party creates about its intentions [33]. It is established through experiences of past actions by others. Conceptionally, in a reputation statement a reputation *source* makes a reputation *claim* towards a certain reputation *target* [34]. Claims may be qualitative (e.g., free form text or media uploads) or quantitative (e.g., normalized scalar values or rank values). The source and target of reputation statements can be any entities or reputation statements themselves. Reputation systems store reputation statements, aggregate them, and transform them into a meaningful form to the user.

Reputation systems recently experienced challenges. Fake reviews threaten the value provided to customers of e-commerce stores [35, 36]. Furthermore, in some business situations, the actors do not have a long track record of business history with collected reputation statements. Examples thereof are especially relevant when micro-SMEs (small and mid-sized enterprises) are involved in business transactions. A family renting out their own house once a year through AirBnB when they are on vacation themselves leaves no opportunity to collect a long track record of reputation statements.

Thus, a new view on trust is needed to accommodate changing business interactions. This thesis proposes a model for trust in business interactions focused on processes. Therefore, concepts from the research field of business process management are primary tools. The next section provides an overview of business process management and how it can incorporate trust.

2.2 Business Process Management

Business process management (BPM) is a research discipline with roots in business administration and computer science. This section establishes the foundational concepts that the later chapters utilize. The provided summary is based on standard literature, especially following the definitions provided by Weske [37].

A *business process* is a set of activities executed jointly to achieve a specific business goal. The discipline of *business process management* "includes concepts, methods, and techniques to support the design, administration, configuration, enactment, and analysis of business processes" [37].

The term business process refers to the enactment of a set of activities in an organization "in the wild". *Process models*, on the other hand, are an abstraction of a business process that can be illustrated visually. Conceptionally, a business process model consists of *nodes* and *edges*. Nodes represent models for activities, events, and gateways in a process. Edges express sequence or message flows. Activities represent units of work in a process. These units of work may be tasks

executed by automated systems, manual tasks, or whole subprocesses. Events illustrate the reaching of a specific point of interest. Examples of that are start, end, or failure events. Gateway elements such as splits or joins express decision points or parallel workflows. Edges are the connection between different nodes. They can represent sequence flows (within one organization) or message flows (across different organizations). Many business process modeling languages enable the user to illustrate different organizations or organizational units using *lanes* or *pools*. Processes with different organizations collaborating in a joint workflow are called *collaborative business processes*. *Process choreographies* are a similar view on cross-organizational collaboration. Choreographies focus on the interaction between independent processes executed by separate organizations. In contrast to choreographies, collaborations interpret inter-organizational collaboration as one process. The subsets of process parts for different organizations are separate subprocesses of the overall process. Section 2.2.1 presents the BPMN modeling language, which has capabilities to model inter-organizational workflows and other more semantically advanced concepts in more detail.

Business process models are the main artifacts in business process management. They can be implemented for either manual workflows with organizational rules and policies or in an automated fashion using software systems. Such systems are called *business process management systems* (BPMS).

In literature, the term business process is often synonymously used to describe a business process model (the "blueprint" for instances) and the process instances derived from the model. This work uses the term business process model and business process instance or case to distinguish the concepts.

2.2.1 Business Process Management Notation BPMN

The Business Process Modeling Notation (BPMN) is a standardized modeling language that exists currently in version 2.0 [38]. Its development was coordinated by the Object Management Group (OMG). BPMN aims to provide a notation that is readily understandable by business users and technical developers at the same time. The following section is based on an introduction by Weske in [37] and the official specification [38]. It provides an overview of the core concepts of BPMN without any claim to completeness.

The core elements of BPMN diagrams can be divided into four categories as seen in Figure 2.4. *Flow objects* (1) include events, activities, and gateways. Events represent a state of interest in the real world. Activities describe units of work. Gateways model split or join behavior. *Swimlanes* (2) represent inter-organizational aspects in a two-level hierarchy. Pools represent organizations, and lanes model subunits within one organization, such as a department. In choreographies and collaborations, each of the organizations is responsible for its own (sub)processes. The interaction between organizations is represented by interaction between different lanes. BPMN uses *artifacts* (3) to show additional information about the business process that is "not directly relevant for sequence flows or message flows of the process" [38]. Artifacts include data objects, groups, and annotations. The last main category of BPMN elements is the set of *connection objects* (4). Connection objects link flow objects, swimlanes, and

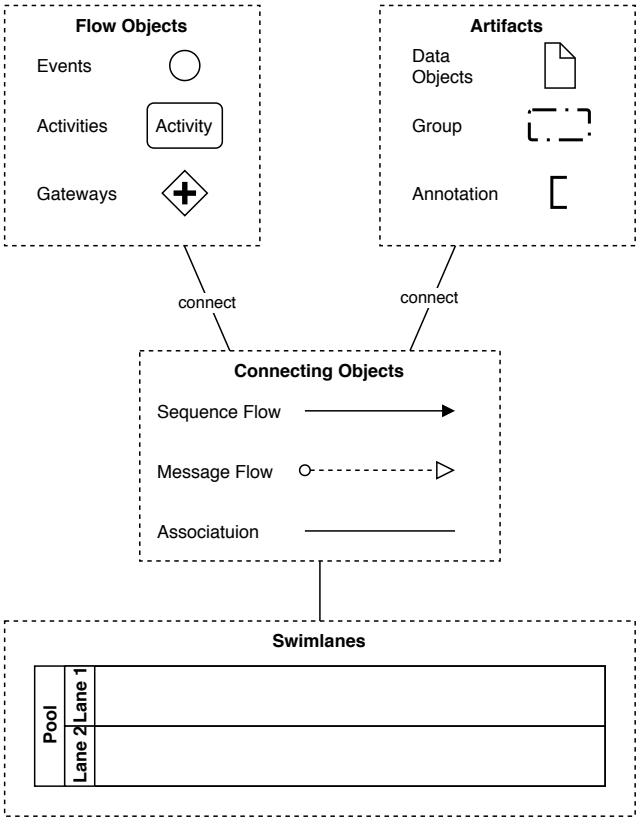

Figure 2.4: Illustration of BPMN 2.0 category elements as defined formally in the standard [38] by OMG. Own illustration.

artifacts. They can be further divided into elements representing sequence flow (within one organization), message flow (between different organizations), and associations between different annotation elements.

Figure 2.5 shows an example BPMN business process model of an apartment rental case using a multi-sided platform. The process involves three collaborators, modeled in three different swimlanes. The tourist wants to stay during the vacation at the flat of the apartment owner. The tourist discovered the apartment using the platform. The process starts at the point where the tourist decided to book. After starting the process, the tourist makes a deposit to the platform provider. The platform provider transfers the money to an escrow account. The interaction between the tourist and the platform is modeled using message exchanges. While the tourist waits for the first day of vacation, the platform provider requests the apartment owner to prepare for the key handover on the first day of the stay. Once the day arrives, the owner hands over the keys to the tourist and communicates the terms of stay. The terms of stay are modeled using the data object artifact attached to the message exchange between tourists and apartment owners. Afterward, the tourist stays at the apartment throughout the vacation period. If something undesired happens, for example, the tourist sets the kitchen on fire, the tourist is required to open an incident and inform the owner. The handling of the incident is modeled in a separate subprocess. At the end of the stay, the tourist returns the keys to the apartment owner. Then, both need to confirm the end of the booking to the platform. Afterward, the platform provider retrieves the money from the escrow account and transfers it

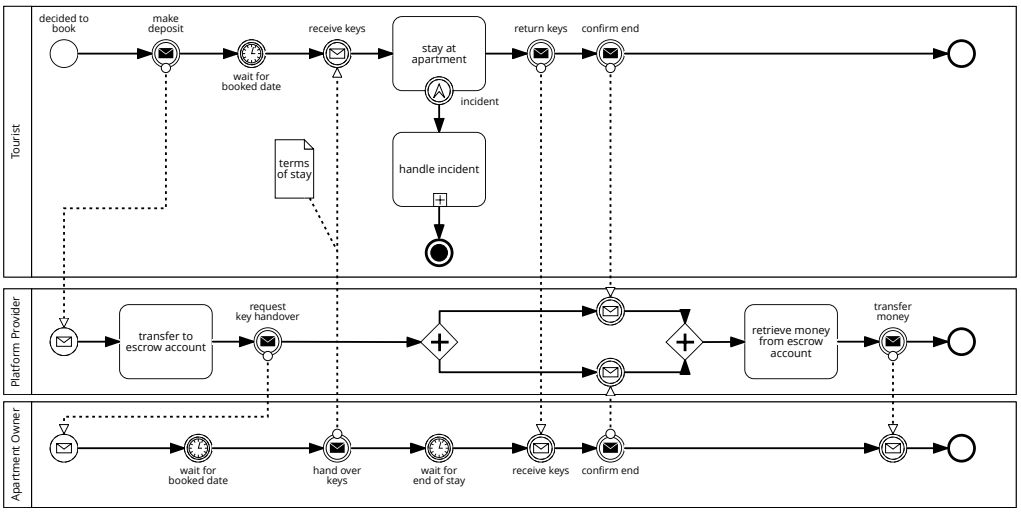

Figure 2.5: An example business process model of an apartment rental scenario using BPMN 2.0.

to the owner as compensation. This marks the end of the process.

Flow objects, swimlanes, artifacts, and connection objects are mere categories of objects in BPMN diagrams. The example above implicitly used specific instances of these process component types. A more detailed overview of the BPMN elements can be found in Appendix A or in [38].

2.2.2 Business Process Management Systems

The previous section introduced the modeling of business processes. This section describes a high level of abstraction of the building blocks of *business process management systems (BPMS)*. Such systems are used to control the enactment of instances of business processes based on their defined process models.

Figure 2.6 shows the high-level architecture of business process management systems. It consists of different components that are modeled as a subsystem of the BPMS. The process engineer (user of the BPMS) uses the subsystem for *business process modeling* to create business process models. This mostly utilizes a business process modeling language, for example, BPMN, as discussed before. The *business process model repository* stores process models. The *business process environment* triggers the creation of a new process instance. The *process engine* is responsible for creating a new process instance based on the process model as stored in the business process model repository. The process engine is the core component of a BPMS in charge of controlling the execution of the process. For the execution of a specific activity instance, the engine calls entities that provide the required functionality. These *service providers* host application services that are used for the business logic of activities. Service providers may be web services or employees of an organization that execute manual tasks.

Business process executions are, from an architectural point of view, distributed by nature. The different components of a BPMS and the external service providers can be represented in each process instance as independent agents.

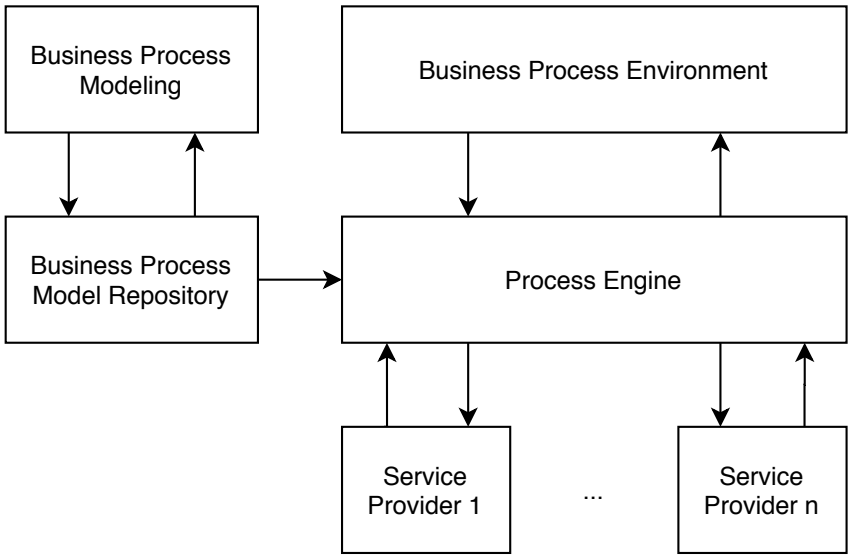

Figure 2.6: High-level architecture model of business process management systems according to Weske [37](p.120)

These interact exclusively by sending and receiving messages. In Figure 2.6 these messages are shown through the arrows.

From an execution view, a BPMS utilizes a process engine for the orchestration of business process instances. The process engine evaluates gateways, executes the process flow, and calls external service providers for certain activities. Distributing the process execution of different activities to separate service providers is especially suited in collaborative processes.

2.2.3 Business Process Management Lifecycle

The development of business process management solutions follows an inherent lifecycle with different phases. The following section provides an overview of the BPM lifecycle presented by Weske [37] or Dumas et al. [39].

Figure 2.7 shows that the business process lifecycle consists of four main phases.

Design and Analysis The design and analysis phase marks the entry point of the BPM lifecycle. The analysis part of this phase conducts surveys on the business processes and their technical and organizational environment. These surveys are used as a base for business process identification, review, and validation, ultimately leading to the representation in business process models.

Configuration The configuration phase marks the implementation of the business process. It follows the design and verification phase of a process model. Therefore, different parts of business processes may be implemented using different tools. For manual activities, the business process implementation may be the distribution of policies and procedures to the employees of that company that need to comply with it for process instances. In case the process part is implemented using a software system, an implementation platform is chosen

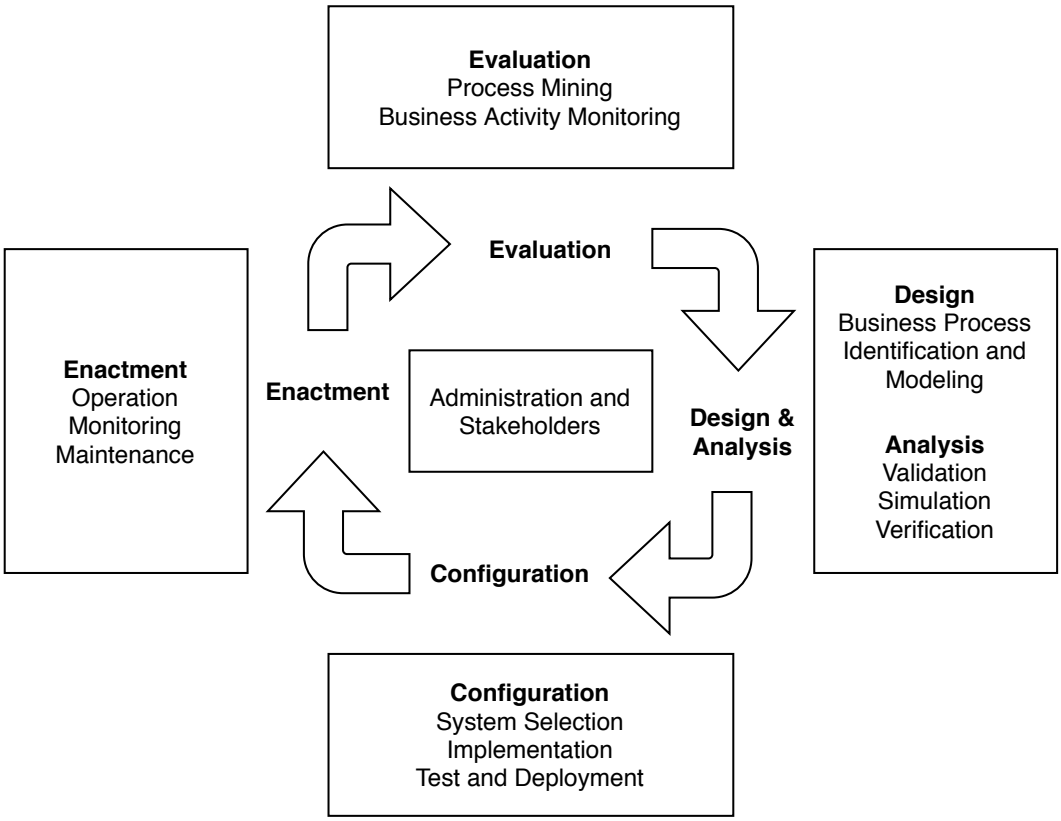

Figure 2.7: The business process lifecycle according to Weske [37], p. 12

during the configuration phase. Either existing software needs to be configured to be used throughout process instances, or new logic needs to be implemented. The configuration phase also deals with technical details such as databases and data storage, transactional properties of data modifications, and integration of different software components that interface each other in the process. Once the system is configured, it is necessary to test the business process implementation. This may include functional and non-functional testing. After the process has been tested, it gets deployed to the previously defined environment.

Enactment The enactment phase follows the completion of the configuration phase. During this phase, business process instances can be instantiated and executed. Thereby the enactment phase marks the actual run time of the business process. The business process management system orchestrates the execution of different business process instances. Therefore, the system follows the process models. During the process execution, a monitoring component visualizes the state of different process instances. Further, detailed information about the execution of different process instances is gathered and usually stored in a log file.

Evaluation The evaluation phase uses data gathered during the process execution and evaluates the processes and their implementations. Typical metrics that

the evaluation phase identifies is the time performance of activities or conformance of the executed instances of the process compared to its model. Therefore, Process Mining [40] as a way to reverse-engineer process models from actual executions is a popular tool.

The relationships between the different phases have a cyclic structure and illustrate the logical dependencies. However, they do not represent a strict temporal order and do not indicate which phase needs to be executed after which other. There is a general consensus on the BPM lifecycle in the BPM research community. While others group the discussed phases differently or rename them (see, e.g., [39]), the main activities stay the same.

2.2.4 Trust-aware Process Design

Trust-aware process design is a relatively new subfield of business process management and has been firstly established by Rosemann in 2019 [9]. The motivation for a trust-centric view on business processes results from a trust erosion experienced towards businesses, government, non-government organizations, and media [41].

Rosemann identifies trust as "the result of a subjective assessment of this uncertainty (belief), and only if sufficient confidence exists will the process be initiated (action)" [9], following concepts established by Sztompka [10]. This definition also shows that trust only becomes relevant if there is uncertainty present, as established in Section 2.1. In his paper, Rosemann examines the research question on how trust can be embedded in the design of business processes. Therefore, the author proposes a four-step approach for trust-aware process design consisting of *identifying moments of trust (1)*, *reducing uncertainty (2)*, *reducing vulnerability (3)*, and *building confidence (4)*. The TAPE framework, which this work proposes in Chapter 3, is built on top of this high-level meta-model by Rosemann and develops the four major steps further.

Step 1: Identify Moments of Trust Trust is needed when a party is about to make a decision influenced by uncertainty regarding another party's action. Rosemann describes when a customer of an e-commerce shop decides whether to hit the "order now" button as such a moment of trust. In this situation, the customer assesses whether she trusts that the payment is processed correctly, the shipment is sent, and reaches the customer's house at the desired time. The customer cannot influence these uncertainties and has to trust in the correct execution.

Step 2: Reduce Process Uncertainty Reducing process uncertainty aims to minimize the probability *that* something undesired happens in the process. Therefore, Rosemann proposes to reduce process variation by reducing operational uncertainty (a), behavior uncertainty (b), and increase visibility (c) to reduce the perceived uncertainty in a process. Reducing operational uncertainty focuses (a) on reducing process variation. Therefore, process engineers can use different

methodologies like Six Sigma [42]. Reducing operational uncertainty aims for a process where uncertainty *practically* disappears. For example, if AI-based medical interpretations of x-ray photographs have a 100% accuracy, uncertainty practically disappears, especially compared to physicians manually interpreting the results. Reducing behavioral uncertainty (b), on the other hand, focuses on reducing process variation that comes from employees involved in the process. Systems, processes, and business rules often provide employees with a certain degree of freedom to perform their allocated tasks. For example, a mortgage broker has to evaluate different mortgage requests. There are usually procedural guidelines on evaluating mortgage requests, but the broker has a certain degree of freedom. Reducing these behavioral uncertainties in a process can be done by automation, clear articulation, and enforcement of rules or incentivization. Increasing visibility (c) is an approach that does not reduce the actual uncertainty of a process. It decreases a customers' perceived uncertainty. For example, a factory with glass walls lets the customer transparently observe the production process. This may decrease the perceived uncertainty for that customer.

Step 3: Reduce Process Vulnerability Usually, it is not possible to eliminate all uncertainties in a process. Therefore, reducing process vulnerability aims to mitigate the impacts *when* something undesired happens. Thereby, it is a reactive strategy, in contrast to reducing uncertainties before manifesting. Vulnerabilities always lead to some kind of cost once they manifest. Thus reducing vulnerability aims to lower the cost that an undesired process outcome may have. For example, if a carrier does not deliver a customer's package within 48 hours, offering an 80% refund on the shipping price is an approach to reduce process vulnerability.

Step 4: Building Confidence Besides reducing uncertainty and vulnerability, building confidence is the last way to improve trustworthiness in processes. Rosemann proposes to add additional sources of trust to enhance the process stakeholder's confidence in the relationship with the previously identified process uncertainties and vulnerabilities. Therefore, six additional sources of trust are discussed. Democratic trust (1) describes trust that comes from the positive experiences of a majority of users who engaged in a process before. For example, the Uber app shows the average estimated time of arrival for the car drivers. This metric is based on other users' experiences and poses a democratic source of trust in the process. Local trust (2) describes that a stakeholder trusts a process because trusted people trust it. This can be described as the "wisdom of friends" who are personally known and context to the anonymous "wisdom of crowds" as gained through democratic trust. For example, if a parcel's sender trusts the process carried out by a certain company because a close friend recommended the company, this describes local trust. Global trust (3) derives confidence in a process only from actors commonly accepted through a high reputation. An example, therefore, is a user who trusts the process of assessing x-ray scans for cancer prevention because a well-known cancer expert assesses with a high reputation. Specific trust (4) describes that confidence in a process comes from

the fact that the execution persons are similar to the user and, hence relatable. An example of that is an online healthcare network where people share their experiences for preventing certain diseases. Trusting suggestions by people who are similar to a user based on their demographics is an example of specific trust. Institutional trust (5) derives confidence from the fact that a certain institution executes the process. For example, a user trusting the German tax-office because it is a well-known institution with no negative experiences is an instance of institutional trust. The last source to build confidence is robotic trust (6). Here, confidence in the process results from the fact that it is automated highly. An example of robotic trust is a user trusting a fully automated navigation system in a car.

Other approaches for Trust-aware Business Processes Apart from Rosemann's concept of trust-aware process design, relatively few publications focused on a structured view on trust from a BPM point of view.

Mohamaddi et al. introduced different concepts to elicit certain users' trust requirements in a process [43]. The concept realizes the identification by consulting the potential users of a software system upfront regarding their trust concerns. Hence, this method describes a top-down approach, starting at the end-user layer and deriving the requirements for the underlying systems from that. In [44], they have further described how to enhance business process models visually with trust-requirements.

Apart from that, research on a trust view on business processes is mostly absent in current literature. However, there are different concepts in other sub-fields of quality [45] or risk-aware business process management [46]. Section 3.2.1 further examines the positioning of trust-aware BPM in comparison to these related concepts. Chapter 3 distinguishes these three BPM sub-fields in detail.

2.3 Blockchain and Distributed Ledger Technologies

The terms blockchain (BC) and distributed ledger technologies (DLTs) have been used in different ways in the past, and the boundaries between the two concepts are somehow fuzzy. Only recently, a de facto consensus on the use of the terms has been established in the research community. To have a common understanding of these terms, this section provides definitions. Afterwards, specific relevant aspects of DLTs are reviewed.

A *distributed ledger system* is "a specific implementation of the broader category of shared ledgers", which let different parties share data records [47]. All data records together represent at any point in time a *global state* of particular pieces of information. State changes can be inflicted by the use of *transactions*. The different parties engaging in the distributed ledger system use *nodes* to interact with the ledger and each other in a *peer-to-peer* network. Between the different parties, *consensus* is established on the accepted transactions and their order. The consensus ensures consistency of the current state across all participants [48]. Distributed ledger systems are typically a "distributed,

cryptographically secure, and crypto-economically incentivised consensus engine" [49]. *Distributed ledger technologies* (DLTs) are the technical foundations enabling distributed ledger systems.

A *blockchain* [15] is a special type of distributed ledger. It is characteristic for blockchains that transactions are grouped into *blocks*. These blocks are cryptographically linked to each other, forming the metaphorical chain of blocks that gives the blockchain its name. Historically, blockchains originate from the finance domain. The blockchain technology's first inception, Bitcoin [15], enables peer-to-peer payments without the need to go through an institutional intermediary, such as a centralized bank. To maintain integrity (e.g., to mitigate the double-spending problem) within the network, Bitcoin uses *digital signatures* and the Proof of Work consensus mechanism. During the evolution of the blockchain technology, a second generation of blockchains have evolved that enable users to execute Turing-complete programming logic called *smart contracts* in a decentralized way. The advance of smart contracts enabled the application of the blockchain technology to other domains such as e-commerce [2], supply chain management [50], and the Internet of Things [51].

Throughout this work, the terms blockchain and distributed ledger technologies are mostly used interchangeably. While there are technical differences, as described before, from the point of view of BPM blockchains and other DLTs offer the same interesting features: an immutable shared storage of the state of particular pieces of information and mechanisms to inflict state transitions in an (practically) integrity-preserving way.

In current literature, many publications provide classifications blockchains and DLTs based on different aspects [52, 48, 53]. The following sections review selected technical properties of blockchains and DLTs that are relevant from a BPM view.

2.3.1 DLT Data Structures

For the implementation of a distributed ledger, specific data structures can be utilized. Blockchain as a data structure is a cryptographically linked chain of blocks. New transactions are grouped into blocks during the consensus process. When conflicts occur, the longest chain is selected by the consensus participants. It is possible to modify this approach to improve scalability [54]. The Greedy Heaviest-Observed Sub-Tree protocol (GHOST) [55] is a different approach, where miners reference competing independently-mined blocks called *uncle blocks*, to increase the probability to be selected as a new block in the main chain.

Another distributed ledger data structure is a directed acyclic graph (DAG) [56]. A DAG allows non-conflicting transactions from uncles to blocks to become a part of the main chain. This is their primary difference to linearly ordered blockchains. Blockchains and DAGs are the main data structures utilized to implement distributed ledgers.

2.3.2 Permission

The *permission* aspect of a blockchain or DLT network defines who can participate in the maintenance procedure of the ledger. This may include aspects like the permission to join the network, initiate transactions, or to verify transactions for the consensus.

Permissionless In a *permissionless DLT*, every entity with access to the DLT can join the network, submit new transactions, and participate in the consensus. Verifiers gather submitted transactions from all participants and add them after successful verification and consensus to update the ledger's state. Examples for permissionless blockchains are Bitcoin [15] or Ethereum (public main-chain) [16].

Permissioned A *permissioned DLT* restricts the set of participants. This may be a restriction regarding the submission of new transactions, a permission to join the network, or a permission to verify transactions. Current permissioned blockchains include Ripple [57] or Monax [58].

2.3.3 Scope

From a system architecture perspective, one important design decision is the *scope* of the DLT [52]. The scope addresses the form of deployment and operations of a distributed ledger. While the permission property of a DLT defines different general levels of participation restrictions, the scope of a DLT describes which entities have write-access to the ledger. Current literature distinguishes primarily between public DLTs, consortium DLTs, and private DLTs.

Public The early inceptions of blockchains were targeted to be primarily cryptocurrencies used for decentralized payments. They can be accessed by anyone on the Internet. This makes them transparent and auditable. In many public permissionless blockchains, scalability is often an issue. Due to the omni--accessibility and modifiability by unknown parties, the DLT network needs strong security mechanisms to ensure the ledger's integrity. Bitcoin [15] is an example of a public permissionless blockchain, while Sovrin [59] is an instance of a public permissioned blockchain.

Private and Consortium Distributed ledgers that are intended for the exclusive use between different organizations are called *private or consortium* ledgers. Only a set of defined organizations can submit transactions and the consortium as a whole has an incentive to ensure the integrity of the network. Hence, consortium DLTs are typically permissioned. Examples include Quorum [60] or Hyperledger Fabric [61].

In current literature, there is no widely accepted consensus regarding the concepts of distributed ledger permissions and scope properties. In the following, this work will use the concepts as described in the definitions above.

2.3.4 Consensus

A *consensus protocol* enables maintaining a distributed ledger of records that all network peers agree upon. In current literature, a plethora of different consensus algorithms for DLTs is present. This section does not aim to provide a complete overview of all consensus protocols. Instead, it highlights the most prominent ones. For more extensive discussion, different surveys provide structured entry points for further reading [62, 63, 64, 65, 66].

Proof of Work In Bitcoin, the Nakamoto consensus utilizes the Proof of Work (PoW) algorithm to mine new blocks of the blockchain [15]. Bitcoin is mostly used in the finance domain. The consensus mechanism was first introduced to prevent the double spending problem of the Bitcoin cryptocurrency. PoW does that by requiring the network peers to solve a mathematical challenge. Technically, this challenge invokes finding a number that satisfies certain complexity criteria. Finding this number is computationally hard, but verifying if a given number satisfies the criteria can be done fast. In Bitcoin, miners compete to find the next block in the blockchain. Thus, at a point in time there may exist different competing versions that are based on the same chain. These different chains are called *forks*. Ultimately, the Nakamoto consensus protocol defines that the longest chain wins. Executing PoW requires a large amount of energy for mining the new blocks. When miners increase their computing power to increase the probability of mining the next block, this yields economic benefits since the miners of a new block receive a reward. The reward is paid in cryptocurrency. The more computational power an entity owns to mine new blocks, the higher the potential yield is. This leads to incentivize miners to maintain a large number of machines, which in turn leads to a significant energy consumption [67]. Different implementations of the PoW algorithm exist including Hashcash [68] (used in Bitcoin) or Ethash [69] (used in Ethereum 1.0).

Proof of Stake In the Proof of Stake (PoS) consensus mechanism, the validators of the next transactions (i.e., the miners in most implementations) obtain the eligibility to create the next block based on their holding of the native digital currency. In Peercoin [70], for example, miners need to prove their ownership of a certain amount of the native cryptocurrency to mine blocks. There is a variety of different PoS protocols, including Tendermint [71] (used in Eris) or Casper [72] (used in Ethereum 2.0). Delegated Proof of Stake (DPoS) is a widespread variation of PoS used in BitShares [73], Steem [74], or Lisk [75]. In DPoS, the holders of cryptocurrency do not claim the right to mine new blocks by themselves. Instead, they delegate the right to other miners and split the rewards. PoS-based protocols are generally more energy-efficient, compared to PoW [76].

Practical Byzantine Fault Tolerance Practical Byzantine Fault Tolerance protocols (PBFT) refer to a commonly used family of consensus mechanisms. A Byzantine Fault describes a condition in a distributed system where components may fail or exhibit arbitrary wrong behavior. Furthermore, the different network components have imperfect information on whether certain distributed system components have failed. The term Byzantine Fault originates from the Byzantine Generals Problem [77]. This allegory describes a situation where the actors in a distributed system must agree on a strategy to avoid a catastrophic failure, while some of the actors are unreliable. A system that is resistant to a certain extent to Byzantine faults is described as Byzantine Fault Tolerant (BFT). In protocols that enable BFT, a node called the *leader* is elected by all other nodes. Non-leader nodes are called *backup* nodes. Any backup node can be elected as a leader in a round of the PBFT protocol. PBFT protocols consist of different phases. Throughout the phases, clients send their proposed transactions to the leader. The leader proposes, validates, and orders them to update the distributed ledger's state database. The backup nodes need to verify the proposal of the leader. PBFT is fault-tolerant as long as less than a third of all nodes act maliciously. Protocols that enable BFT usually scale to a very limited extent. The larger the number of nodes participating in the consensus, the longer it takes to reach consensus. Thus, PBFT protocols are often used in private permissioned distributed ledgers, such as Hyperledger Fabric [61] or Stellar [78]. The core of Tendermint [71] is also a PBFT protocol. In general, BFT-enabling ledgers offer strong consistency guarantees and low latency. This comes with the price that only a small number of participants can be part of the consensus [54]. It is also required that the participants agree on the list of peers in the network. This makes PBFT a popular choice for private permissioned ledgers.

Other Consensus Mechanisms Apart from the aforementioned three most widely spread consensus mechanisms, many others are currently utilized for DLTs. The *Proof of Authority* (PoA) consensus mechanism does not require the participants to solve mathematical problems [79]. Instead, a limited subset of peers is authorized to create new blocks for the Distributed Ledger. The inclusion of a node as an authority is subject to voting. Kovan is an implementation of a PoA distributed ledger utilizing the Ethereum Virtual Machine (EVM) as its core [80]. The *Proof of Burn* (PoB) consensus mechanism requires miners to send "coins to an unspendable address in exchange for a probability to find Proof of Burn blocks and get block rewards regularly" [81]. The more coins a peer burns, the higher the chances to be the miner of the next block. Slimcoin [82] implements the PoB mechanism in the use case of a cryptocurrency. In a network that uses *Proof of Importance* (PoI) as a base for the consensus mechanism, the eligibility to create new blocks is determined through an importance score. This score symbolizes the importance of certain peers in the topology of the network. Nem is a distributed ledger that utilizes PoI [83]. Accordingly, Nem employs the NCDawareRank algorithm, a modification of the PageRank [84] algorithm to calculate the importance score of peers in the network.

2.3.5 Smart Contracts

The initial use cases of DLTs focused on the finance domain. The goal of the consensus in finance is to agree on a validated and ordered list of transactions that can be traced to validate the current balance of the owning accounts. Technically, this can be regarded as a state change of particular pieces of information, i.e., the balance of the different accounts. The allowed state changes in the finance domain are restricted. One user can transfer monetary value from her owned account to one or more other accounts. All peers in the ledger's network validate the transactions to prevent a user from spending money that she does not have.

However, the same principle can be applied to state changes of *arbitrary* pieces of information through *smart contracts*. Szabo introduced the term in the 1990s. He defines a smart contract as "a computerized transaction protocol that executes the terms of a contract" [85]. The concept has the goal of automatically enforcing contractual agreements by minimizing malicious and accidental exceptions from the terms. Thus, smart contracts aim to minimize the need for trusted intermediaries. Wood and Buterin adopted the concept of smart contracts in the Ethereum blockchain [16]. Technically, a smart contract is a piece of programming logic encapsulated in code. The logic defines how different information may be modified. Smart contracts may define that an escrow can only release payments after several parties agreed or that a package with physical goods may only be handed over to a certain actor. The code is submitted to the distributed network and all its peers. State changes are triggered through transactions. Once a peer validates a transaction that triggers a smart contract function, the peer executes the logic locally and updates the state database. During the consensus phase, the other network participants validate the transactions, compare their local state, and propagate the changes to their copies of the state database. Hence, the state change propagates through the network. With the validation and the consensus protocol, the validity of the smart contract execution is enforced.

Blockchain systems that support the execution of arbitrary smart contracts are called *smart contract platforms*. Ethereum [16] is one of the most prominent smart contract platforms. It enables to define smart contracts in different programming languages, such as Solidity or Vyper [86]. Other smart contract platforms include Hyperledger Fabric [61], Quorum [60], Corda [87], or Polkadot [88].

2.3.6 Scalability

With the growing adaption of distributed ledger technologies and an increasing number of users, scalability issues arise [53]. Transaction throughput and confirmation latency are two major performance metrics of DLTs that suffer from scalability issues.

In current public distributed ledger systems, these performance characteristics have not reached a satisfactory level at the time of writing [89]. Due to the decentralized nature of DLTs, this challenge is hard to solve. Informally, DLT scalability issues have been set in context to the degree of decentralization and security. These three dimensions are referred to as a trilemma [90]. The

trilemma claims that scalability, security, and decentralization in a distributed ledger cannot perfectly co-exist. This circumstance is comparable to the CAP theorem in traditional distributed systems [91].

In current literature, different ways to solve the scalability issues of distributed ledgers are present. The solution approaches focus on blockchains as the currently most prominent data structure used as a distributed ledger [92]. Scalability solutions can be classified regarding their hierarchical structure in layers. Layer 1 solutions concentrate on the on-chain design of the blockchain. This includes the utilized consensus mechanism, the structure of the blocks, and the overall structure of the main-chain. Layer 2 solutions intend to reduce the computational burden on the main-chain by utilizing off-chain approaches.

The first Layer 1 solutions approached the scalability problem by modifying the block size. Bitcoin Cash, for instance, modifies the Bitcoin protocol with a larger block size to accommodate more transactions in the same block [93]. The compact block relay aims to save bandwidth in the peer-to-peer network by compressing blocks [94]. More recent Layer 1 solutions utilized improved consensus mechanisms to address scalability [95, 96, 97]. State sharding techniques aim to distribute the current state database between different subsets of participants [98, 99].

Layer 2 solutions shift a number of transactions to an off-chain computing environment to reduce the load on the main chain. This includes payment channels in the finance domain [100] or generalized state channels for arbitrary smart contracts [101]. More recently emerged cross-chain solutions also play an important role in Layer 2 scalability solutions. For instance, Cosmos aims to connect multiple independent ledgers to establish an integrated network [102].

Layer 1 and Layer 2 scalability solutions make compromises regarding the fundamental properties of decentralization and security. When incorporating such solutions in an application, these implications need to be considered.

2.3.7 Privacy

The second major challenge that distributed ledgers currently face is privacy. DLTs draw their strong security from the underlying data encryption, time stamping, and incentive mechanism, instead of relying on a trusted third party [103]. Trust can be established between the nodes in the network by verification and transparency of past transactions [104]. On the other hand, this transparency is the source of many privacy-related challenges.

One of the major architectural decisions that influence privacy is whether to employ a permissioned or permissionless ledger. A private permissioned ledger only allows authorized entities to access state data. In public permissionless networks, on the other hand, every entity can join and leave the network freely [105]. Peers have access to all state-related data to verify the integrity of the current state information. The disclosure of transaction content in public permissionless ledgers may lead to crucial privacy leaks. The identifiability of a user by its pseudonymous address leads to a variety of different risks. Third parties may be able to analyze the transactions of a certain user. Also deeper analysis of transaction rules is possible. In the past, different approaches to create an

association between the user and her used addresses aimed to reverse-engineer the identities behind the pseudonymous identifiers [106, 107].

The transparency of permissionless DLTs may result in the misuse of user data. For instance, competitive enterprises or individuals can benefit from analyzing transaction data or obtain sensitive user-related data to gain a competitive advantage. However, these privacy issues are not limited to permissionless ledgers. Even in permissioned consortium setups, privacy needs to be considered. For instance, in a network where direct competitors have peers and participate in the consensus, the same privacy issues regarding competition as in permissionless ledgers arise.

Currently, a plethora of different approaches to solve privacy issues in distributed ledgers exist. Later in this thesis, the privacy issue will be extensively revisited in the context of DLTs for trust-aware collaborative business processes. For a further overview on the topic, see surveys such as [108, 109, 110].

2.3.8 Distributed Ledgers as a Part of Software Systems

Distributed ledgers have been identified as a tool to foster trust in software applications with multiple involved parties [4, 111]. Yet, different interpretations of how a distributed ledger specifically improves trust in such applications in detail co-exist. Current literature focuses mostly on blockchain as a distributed ledger data structure. In the finance domain, blockchain is often described as a "trustless" system [112]. This notion refers to the fact that no trusted third party or intermediary is required to exchange financial assets. Blockchain is often phrased as a substitute for trusted intermediaries in the industry and IoT context. But in this domain, the ledger is said to establish trust rather than making the application trustless [51]. Presenting blockchain as a tool to build or increase trust in collaborative business processes [113] has also seen extensive usage across existing literature.

DLTS for dApps Distributed ledgers technically enable *decentralized applications* (dApps). Antonopoulos and Wood define a dApp as "an application that is mostly or entirely decentralized" [86]. Decentralization refers to the degree of distribution of political power rather than to physical distribution. According to Antonopoulos and Wood, dApps create three main advantages compared to centralized approaches:

- **Resiliency**: In a dApp, business logic is controlled by smart contracts. Its execution is managed by the peers of the distributed ledger smart contract platform. Unlike applications that are deployed on a centralized server, dApps are more resilient. As long as not all network peers are offline, the decentralized execution environment has no downtime. It is available as long as the platform is operating.
- **Transparency**: The distributed ledger that enables a dApp allows every party with access to inspect the code and verify its correctness. Each interaction with the dApp through transactions is stored in the ledger as long as the network exists.

- **Censorship resistance**: As long as a user has access to a node of the network, the user is able the interact with the dApp. Centralized access control is not necessary in public permissionless networks. Private consortium networks may limit this property. No entity can alter the code of a smart contract once it is deployed on the network.

This definition implies mostly a public permissionless nature. The term dApp is used for such open networks in the majority of cases. Decentralized finance (DeFi) is the application domain that currently utilizes the term dApp most frequently [114]. However, distributed ledgers do not necessarily require a public permissionless network to enhance trust.

DLTs as Software Connectors A ledger can be used in several parts of a distributed system architecture [115]. Xu et al. conceptualized the role of distributed ledgers as a software connector of different systems [116]. Software connectors are foundational building blocks in complex software interactions [117]. The term connector refers to an interaction mechanism for the components of the software. One example of a software connector is a middleware [118]. A middleware connects to components that interact with it. Connectors are crucial elements in distributed systems. They achieve desired system properties such as performance, reliability, or security. The connectors' capabilities can be regarded as interaction services that are mostly independent of the functionality of the interacting components [119]. The services that a software connector provides may be classified into four categories. Communication services (1) transfer data between components. Coordination services (2), on the other hand, transfer control among the components. Conversion services (3) allow adapting the interactions between components to be compatible with each other. Facilitation services (4) support and optimize the interaction of the different components.

A distributed ledger can be utilized as a complex, network-based software connector. It provides communication, coordination, and facilitation services. Communication is realized through the ledger's native communication protocol that is used for the consensus. The coordination between different components is executed through transactions, smart contract invocations, and oracles that observe the ledger from the outside [120].

Distributed ledgers as software connectors are especially useful for coordinating different software components from separate organizations. Thus, the software components are used to execute parts of business processes. The next section will highlight the utilization of DLTs from a BPM perspective.

2.4 Distributed Ledgers for Business Process Management

Business process management is focused on the design, execution, monitoring, and improvement of business processes. Software systems are often utilized to support the enactment and execution of processes. In the past, BPM has been extensively used by companies to coordinate, improve, and automate

intra-organizational processes (within one organization) [37]. Yet, for inter-organizational processes (across different organizations), other challenges arise. Process orchestrations and choreographies need to be jointly designed. Coordination and real-time integration of processes require mutual trust. These challenges pose obstacle for efforts of integrating inter-organizational processes tighter.

In current literature, different authors identified blockchain and distributed ledger technologies as an innovation that has the potential to drastically change the environment of inter-organizational business processes [4, 121]. Weber et al. conceptualized the principles of using distributed ledgers for business processes [122]. In their work, they claim that "large parts of the control flow and business logic of inter-organizational business processes can be compiled from process models into smart contracts which ensure the joint process is correctly executed" [122]. The internal (sub-)processes in an inter-organizational process can connect to these smart contracts through trigger components. Thus, the trigger components act as a bridge between the ledger and the different software components of separate organizations.

Mendling et al. state that inter-organizational processes that utilize distributed ledgers as described before can remove traditional barriers of the deployment of such processes in three different ways [4]. A distributed ledger can serve as an immutable track record of messages (1). This allows process participants to review the history of messages in a cross-organizational process and pinpoint the source of an error. Therefore, all necessary state information needs to be stored on the ledger. The state-changing messages need to be implemented as transactions on the DLT. Smart contracts can offer process coordination and monitoring from a global viewpoint (2). For instance, smart contracts can enforce message exchanges to be authenticated and authorized for different roles. Last, distributed ledgers can ensure the integrity of data consumed in the process (3). By encrypting and hashing signed data objects, parties that own the raw data objects can validate the lineage of data objects. These three approaches illustrate how distributed ledgers can support organizations to implement and execute business processes across organizational boundaries. DLTs enable such processes while not requiring any trusted third party. Hence, they are a completely decentralized approach.

Following the mission paper by Mendling et al. [4], recent information systems research explored the capabilities of distributed ledgers in different activities in BPM lifecycle [39]. Garcia et al. surveyed recent publications and classified them according to seven BPM life cycle activities [123]. These seven activities can be related to the four major phases in BPM projects as discussed in Section 2.2.3. Their empirical analysis showed that current research on distributed ledgers for BPM is primarily centered on implementation and execution-related activities. 50% of all surveyed publications could be associated with these activities. Two other BPM activities have seen more comprehensive research, while the remaining ones were rather infrequently subject to research in the context of DLTs and BPM. About a third of the survey papers were concerned with discovery and modeling activities and 16% with analysis. The following sections provide a

short overview of the current literature classified by their activities in the BPM lifecycle.

Process Identification Process identification is a part of the design and analysis phase in the BPM lifecylce. It aims to provide a high-level description of organizations and their processes. The overall goal is to link the business strategy with the improvement of processes [39]. Process identification is usually a part in the BPM lifecycle that is independent of any technology. Thus, in this phase, strategic decisions need to be aware of the existence of DLTs and their potential, but concrete actions regarding DLTs are not part of this activity [123].

Discovery and Modeling Process discovery and modeling activities are part of the design and analysis phase in the BPM lifecycle. Typically, these activities are concerned with capturing the as-is process. They utilize techniques such as interviews, Process Mining, or walkthroughs. The goal is to obtain a process model to be used as a foundation for further BPM activities.

Process Mining is a popular tool for discovering the as-is process. Several publications proposed approaches to perform Process Mining on process logs of distributed ledgers. Müller and Ruppel showed in an empirical study how to extract process logs from large-scale blockchain networks like Ethereum [124]. Their study illustrated how Process Mining can be utilized to analyze the evolution of processes executed on a public ledger to draw macro-level insights on the platform itself. Klinkmüller et al., on the other hand, discussed how process logs extracted from distributed ledgers can be analyzed in the context of a specific inter-organizational process (micro-level) [125].

In the current BPM and DLT literature, there are several approaches that enable modeling the role of a distributed ledger in inter-organizational business processes. Yuanyuan et al. propose a framework to model transitions between different participants in a process [126]. Their framework includes a modeling language that defines a sequence of states in a process. The states connect to each other with transitions. Ladleif et al. proposed to extend BPMN 2.0 choreography diagrams with the semantics of distributed ledgers [127]. A BPMN 2.0 choreography diagram models the interactions between two or more organizations that aim to achieve a common business goal [38]. The focus of these types of BPMN diagrams is the interfaces and communication between the process participants. From an outside view, BPMN choreography diagrams represent business agreements (or contracts) and the expected behavior of the process collaborators without showing the details of the executable parts. Falazi et al. [128] extend plain BPMN 2.0 diagrams with a set of elements specific to distributed ledgers. Their approach specified explicitly how the lifecycle elements of a transaction to a DLT could be expressed in a process model.

Analysis Process analysis is a part of the design and analysis phase in the BPM lifecylce. This activity is concerned with assessing the current process performance and problems. The outcomes of this analysis activity is the base for process improvements or complete process (re-)designs.

Regarding the performance of business processes, it is possible to apply Process Mining with the perspective on non-functional attributes like execution time and extend the approaches in [124] and [125]. Yet, for inter-organizational processes with DLTs, additional non-functional attributes can be measured [129]. The time until a transaction reaches finality or a transaction not being included in the state of the ledger are events that need to be considered in the analysis phase in the BPM lifecycle.

Implementation and Execution Process implementation aims to implement and automate a process, often with the use of information systems technology. Business Process Management Systems (BPMS) [37] are software systems capable of executing processes according to their predefined models.

In inter-organizational business processes, the process participants traditionally have their own business process engines. Interfaces between the different parts of the process carried out by different organizations were traditionally coordinated in a peer-to-peer fashion. López-Pintado et al. present a distributed blockchain-based business process engine called Caterpillar [130, 131]. Caterpillar stores process model definitions in smart contracts. Therefore, it uses an external compiler that translates BPMN to solidity smart contracts. The process engine enables the instantiation of business processes and their execution in an on-chain execution environment. From an architecture point of view, the collaborators of the inter-organizational process all need the permissions to access the blockchain. They connect their own private (sub-)processes through oracles to the decentralized process engine. In its reference implementation, Caterpillar uses the Ethereum blockchain and Solidity smart contracts. But in general, the same principle can be applied to other distributed ledgers. Extensions of the principles of Caterpillar also enable on-chain authorization in the process through role-based access control [132, 133]. Sturm et al. propose an optimized implementation of the same principle that consists of merely 100 lines of Solidity code which also includes the possibility to validate permissions and process conformance [134]. Yet this approach includes some faults. Another instance of a blockchain-based model-driven software engineering tool has been introduced by Tran et al. [135]. This approach helps to implement a process in smart contracts.

Monitoring The process monitoring activity collects execution data of processes. In an inter-organizational business process, collaborators can visualize this data. The visualizations can serve as a base for decision making or compliance control. When incorporating distributed ledgers into business processes, new challenges arise. In a traditional centralized BPMS it is only needed to contact a single BPMS to retrieve data to monitor. Most BPMS provide simple interfaces with pre-aggregated data for simple integration. When the BPMS is implemented in a decentralized fashion with a distributed ledger, retrieving process-related data becomes more challenging [136]. The on-chain process-data is usually very minimal and encrypted or hashed. The plain data is stored at the

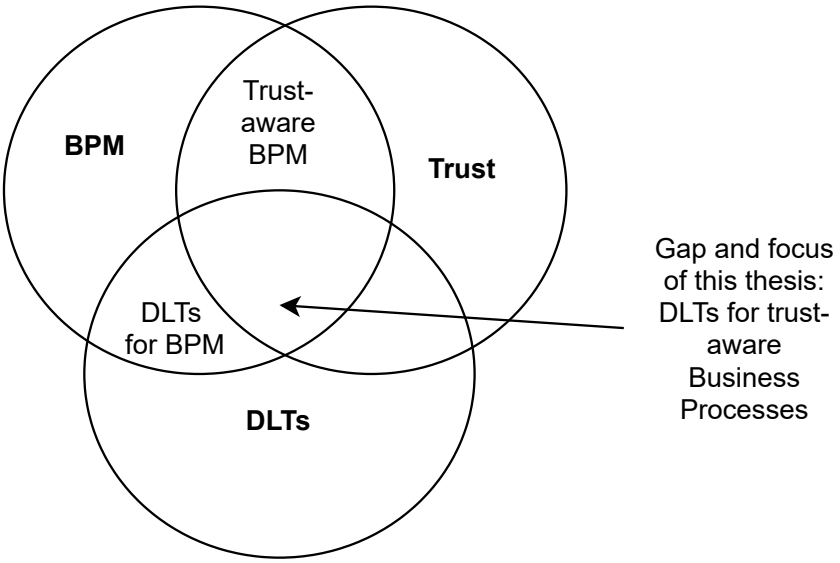

Figure 2.8: Conceptual research gap that this thesis aims to address.

local BPMS of the different process participants. This leads to a fragmentation of the data sources needed to monitor the process [4].

In current literature, the issue of monitoring collaborative business processes with DLTs has remained mostly untouched. Klinkmüller et al. propose the Ethereum Logging Framework (ELF) [137]. ELF allows its users to define relevant data in a distributed ledger for a specific use case. To generate value out of logs, they have to be transformed and formatted. The rules for the transformation and formatting process are also stored in the manifest. The validator component of the ELF can support the users to analyze user-defined manifests for specification errors semantically. The extractor component of the framework retrieves data from an Ethereum node. Therefore, the extractor iteratively traverses the manifest. ELF's last component, the generator, derives cost-efficient logging functionality from the manifest that can be attached to smart contracts. ELF is a minimal framework tied to Ethereum. At the time of writing, another work focusing on a general blockchain logging framework with support for other blockchains is in progress[1].

2.4.1 Research Gap

The previous sections highlighted the most prominent research streams of BPM and distributed ledgers. However, the current research on trust-centered aspects in BPM is nascent [138]. Section 2.3.8 highlighted different relationships between distributed ledgers and trust in different domains. The interpretation reaches from blockchain as a "trustless" system [112] to a tool to increase [113] or enable trust [51]. The relationship of DLTs and trust in the context of a business process has to the time of writing rarely been explored on a fundamental level.

This thesis aims to fill this gap and provide a conceptualization of trust in the context of a business process. Based on this foundation, a new method to analyze

[1] see https://github.com/TU-ADSP/Blockchain-Logging-Framework

trust and mitigate trust issues (TAPE) is introduced. The trust-related concepts are in general domain-independent. This thesis explores deeper distributed ledgers and their relationship to trust in a business process.

Figure 2.8 shows the research gap this thesis aims to fill. The Venn diagram shows the scientific contributions are located at the intersection of BPM, trust, and DLTs.

3 Trust-aware Business Processes with Distributed Ledger Technologies

Trust is an essential part of inter-organizational business collaborations. However, to conceptualize a highly informal notion of trust in concrete business processes is a challenging endeavor that has been mostly absent in current literature in business process management, as discussed in Chapter 2. The following sections establish a model for trust in business collaboration is needed, which is called *TRUB*. TRUB is the foundation of a novel method to trust-aware process engineering, which is called *TAPE*. The TAPE method poses the primary artifact to analyze trust and mitigate trust issues systematically. As the overall modeling approach, business process management is utilized. The subsequent sections examine how distributed ledger technologies can serve as a tool to mitigate trust issues.

3.1 Methodology

Methodologically, this chapter is structured as illustrated in Figure 3.1. This work aims to address the overarching research question on how business collaborations can be systematically analyzed regarding their trust issues in an automated fashion and how these trust issues can be mitigated by using technological innovations. To restrict the field of innovation in general, distributed ledger technologies as a tool to improve trust issues in collaborations are the main focus of this work.

Two hypotheses result from a deeper analysis of the research question.

H1 Mitigating trust issues in business collaborations can be supported by a process-centered design method.

H2 Distributed ledger technologies can be used in different ways to mitigate trust issues in business collaborations.

The methods and concepts introduced in this work follow the design science paradigm for information systems research [19]. Since design science is a problem-solving research paradigm, it is necessary to formulate the hypotheses as problems that can be solved with new artifacts.

Rephrasing hypothesis H1 as a problem to solve yields P1:

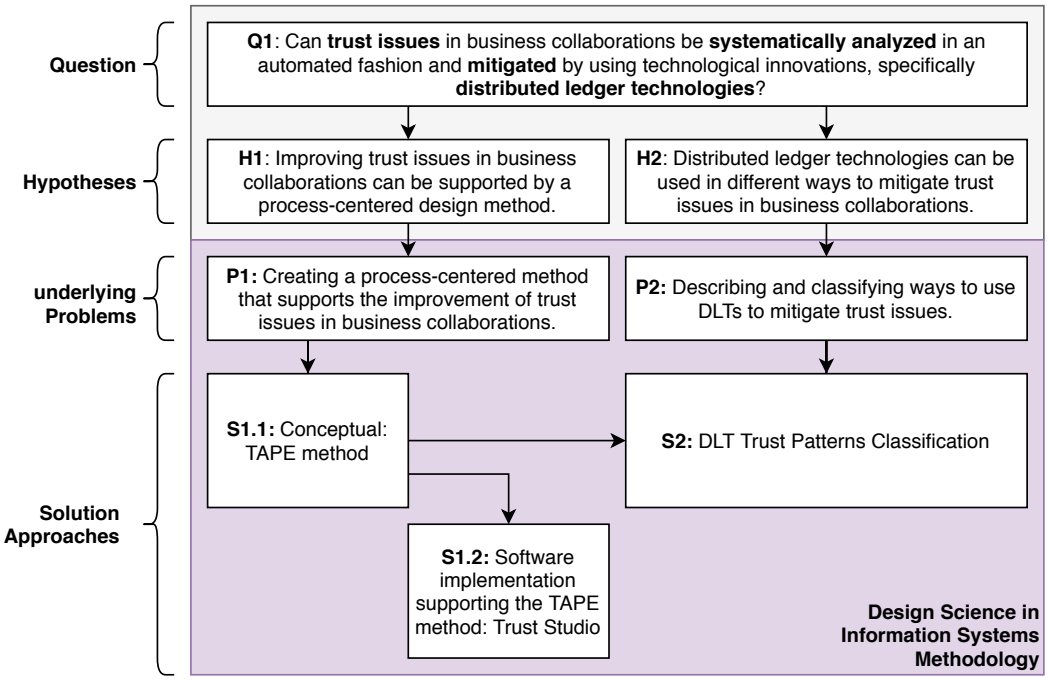

Figure 3.1: Methodological outline of concept and design.

P1 Creating a method that supports the mitigation of trust issues in collaborative business processes.

Find a method that supports the mitigation of trust issues requires a model for trust in business collaboration. Therefore, business process management as a paradigm in model-driven software development serves as a starting point. A business process model can describe a business collaboration's underlying activities and interactions abstractly and formally. The TAPE method adds a novel trust layer to a business process model. This trust-centered process model is the main artifact for the trust analysis and the mitigation of trust issues that the TAPE method proposes in its three-step approach (see Section 3.5). Hence solution S1.1 describes the solution to problem P1. Solution S1.2 introduces Trust Studio as a software artifact to support process engineers and analysts applying the TAPE method.

S1.1 Conceptual method: TAPE

S1.2 Implementation of TAPE: Trust Studio

Hypothesis H2 asserts that distributed ledger technologies can be used in different ways to mitigate trust issues in business collaborations. Rephrased as a problem for the design science approach of this work leads to P2.

P2 Describing and classifying ways to use DLTs to mitigate trust issues.

The solution to problem P2 utilizes the foundational trust model used in TAPE to propose a classification of different distribute ledger trust patterns for collaborative business processes.

S2 Distributed ledger trust patterns classification

The following sections first cultivate the research environment of trust-aware business process management (Section 3.2). This helps to categorize this work's contributions in the BPM landscape. Afterward, the solution approaches are presented in the remaining sections.

3.2 Trust-aware Business Process Management

Trust has been a fuzzy concept throughout the majority of past information systems research. Rosemann [9] introduced the term *trust-aware process design* based on recent trust erosion that could be observed from and between businesses, governments, non-government organizations, and media alike [41]. This study has shown that despite a good economic situation[1] this development occurred in large majorities of the population worldwide. Rosemann argues that business processes are essential artifacts in the operations of businesses, governments, NGOs, and mass media. Thus, the analysis of trust needs to be coupled with business processes.

In the BPM research community, a practice called *x-aware business process management* is the common approach to extend the core BPM methodology with *other* objects or phenomena in a wider organizational context [139]. Recker states that "awareness is generally defined as a state of consciousness in which we perceive and recognize the relevance of a certain object. This means that as individuals, awareness refers to our ability to sense objects and cognitively react to them." [139]. Different instances of x-aware BPM have been proposed in the past. These include but are not limited to context-aware BPM [140], risk-aware BPM [46], cost-aware BPM [141], quality-aware BPM [45], and sustainability--aware BPM [142].

Following the practice of the x-aware BPM pattern, this work proposes to use the term *trust-aware business process management* to extend the traditional BPM with awareness for uncertainty, trust, and related aspects.

Definition 1: Trust-aware BPM
Trust-aware business process management (TA-BPM) is a BPM sub-field in which people use various methods to discover, model, analyze, and improve uncertainty and other trust-related concepts in business processes.

This definition reflects that the notion of *uncertainty* is strongly related to trust. Trust is only needed if uncertainty is present in a process, as commonly agreed upon in trust-related literature [22, 143, 144]. The relationship of trust and uncertainty is essential and will be discussed in more detail as a part of the TRUB trust model that builds the foundation for the TAPE method in Section 3.4.

Figure 3.2 shows how the high-level architecture model of a business process management system can be extended with a trust-aware layer. Modeling business processes requires an additional part for modeling trust. During the enactment of a business process, the services that different providers execute need to be implemented in a trust-aware fashion.

[1] the study dates before the COVID-19 pandemic

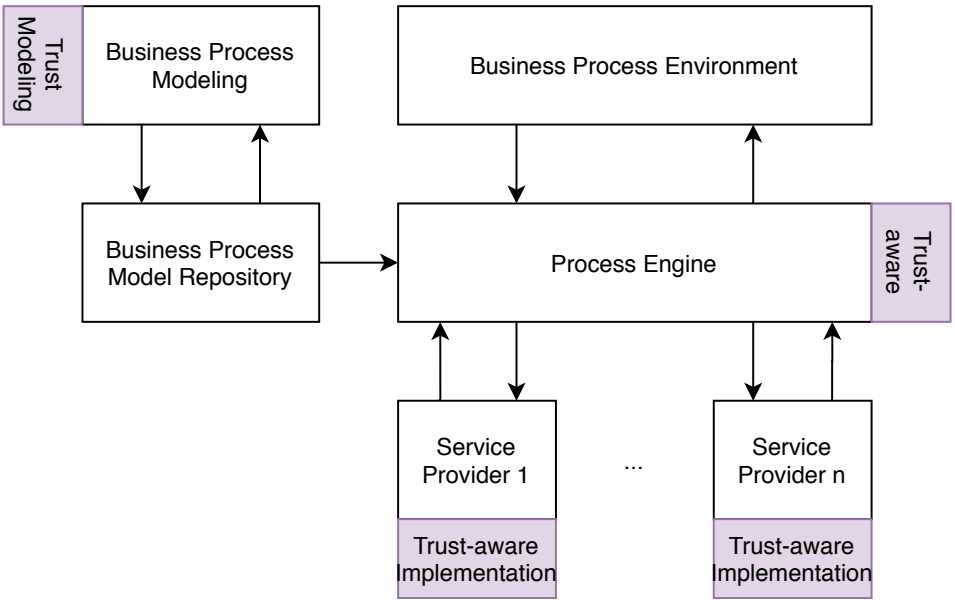

Figure 3.2: Extension of the high-level architecture model of business process management systems according to Weske [37](p.120) with a trust-aware part.

3.2.1 Trust-aware BPM in the BPM Landscape

With the field of trust-aware BPM established, this section locates it within the broad BPM landscape and discusses its differences to other x-aware BPM approaches. Therefore, the closest adjacent fields are quality-aware [45] and risk-aware BPM [46]. Table 3.1 compares the main differences between the three different viewpoints.

Quality-aware BPM focuses on the performance of a process and the quality attributes of artifacts related to it. For instance, whether the process outcome fulfills the process stakeholders' quality requirements is part of a quality-aware approach. Quality-related attributes are, for example, reliability or usability. Different metrics for quality attributes exist. The reliability of a system, for instance, can be measured in downtime. Quality-aware process management influences all phases of the BPM lifecycle. In the design phase, a process engineer can include quality-assuring activities in the process. The implementation phase needs to be aware of the quality-related aspect to ensure the implementation enables executions that match the quality requirements. It is possible to monitor quality attributes of process artifacts during the execution phase. They can be analyzed in the evaluation phase.

Risk-aware BPM mainly focuses on *threats* to a process in a quantifiable way. It has its center on the probability of undesired phenomena in the process. Threats to a process need to be objectively quantified for the inside of the organization that executes the process. That stands in contrast to a quality-aware view. The quality of an outcome of a process can be analyzed without any view of a process's internals. For threats and risks, on the other hand, such an internal view is necessary. Attributes related to risk-awareness are often related to security or safety. Threats to these attributes need to be quantified in a way that enables calculating a probability for them. Threats, their probability,

	Quality-aware BPM	**Risk-aware BPM**	**Trust-aware BPM**
focus of view	process performance and outcome	threats and their probability	uncertainties and their impacts
main concepts	quality attributes	threats, probability and impact	uncertainties
quantifiable	yes	yes	no
perspective	subjective	objective	subjective
involvement	external or internal	internal	external or internal
phase in BPM lifecycle	all	all	design and implementation phase

Table 3.1: Comparison of risk-aware, quality-aware and trust-aware BPM

and their impact once they manifest are the main concepts subject to risk-aware BPM.

Trust-aware BPM is strongly related to both quality-aware and risk-aware BPM and partially overlaps with the two subfields. The focus of trust-aware BPM is concerned with uncertainties and their impacts from an external view on the process. The subject of uncertainties are trust concerns. Some of them are rooted in the field of (information) security, including but not limited to integrity, confidentiality, and non-repudiation. These attributes can also be observed in risk-aware BPM. Also, quality-related trust concerns such as process performance or resilience can be subject to a trust-centric view of business processes. Even though the subjects of trust-aware BPM overlaps with the two adjacent subfields, *how* trust-aware BPM deals with them methodologically differ. In quality-aware BPM, quality attributes are often a *promise* from one process collaborator to another. A trust-aware point of view does not focus on the promise itself but rather on its uncertainty and how different actors address this uncertainty. Risk-aware BPM needs to quantify threats statistically so that probabilities can be assessed. A trust-centric approach, on the other hand, enables the observation of uncertainties in a process that cannot be assessed sufficiently quantifiable from an outside perspective. Also, trust does not distinguish between different uncertainty levels in terms of probability. A trustor either trusts a trustee for an activity in a process or not. In risk-aware BPM, this would mostly not be a boolean assessment, but rather a probability on how likely the trustor estimates that the trustee will execute the activity as intended. Thus, trust-aware BPM is also suited for situations where only assessments from the outside can be made. This is especially common in collaborative business processes.

To illustrate better the difference between the subfields, the following three example sentences show typical questions.

- Quality-aware: "Will the output of activity A fulfill my quality requirements?"
- Risk-aware: "How high is the probability that activity A fails?"
- Trust-aware: "Do I trust organization O, that they uphold the confidentiality agreements of activity A?"

Thus, trust-aware BPM has an overlap with quality-aware and risk-aware BPM but offers a different *perspective*.

3.3 The TAPE Method

This work proposes TAPE (trust-aware business process engineering) as a method that helps to analyze and mitigate trust issues in business processes. Thus, it is specific method in the universe of trust-aware BPM. A high-level overview of TAPE can be seen in Figure 3.3[2].

TAPE is a three-step-approach that builds on a fundamental trust model for collaborative business processes (TRUB). This work introduces TRUB as a novel trust model in the context of business processes. It enables its users to describe uncertainty regarding specific trust concerns and locates them within the elements of a process model. The principles of TRUB are essential for the three steps of the TAPE method. Section 3.4 discusses TRUB in detail.

The primary goal of Step 1 in the TAPE method is to *identify trust problems* in a business processes. Therefore, its process model is used as an input together with a set of *trust policies*. These policies formally describe the trust tolerance profiles of different process stakeholders. The main activity of Step 1 in TAPE is an automated identification of relevant trust problems through *Trust Mining*. This novel method traverses a process model, annotates uncertainties, and analyzes relevant issues for different stakeholders based on their trust policies. Trust Mining is discussed in Section 3.5 in detail. Trust Mining produces *trust statistics* as an output for the different personas that help process engineers and analysts to comprehend uncertainties and trust relationships in the process. Also, it produces a list of relevant trust issues for different process stakeholders (called trust personas). These relevant trust issues are forwarded to Step 2 of TAPE as an input.

Step 2 aims to (re-)design a process in a trust-aware fashion based on the relevant trust issues for different trust personas. Thus, this step aims to *mitigate trust issues* by modifying the process with *trust patterns*. The output of Step 2 is an abstract trust-enhanced process model that needs to be implemented. Section 3.6 deals with Step 2 in detail.

Step 3 takes an abstract trust-enhanced process model produced by Step 2 and *implements* a trust-aware process. Therefore, technology-specific context and organization-specific context need to be taken into account. Step 3 needs to implement fully automated parts of the process, executed by one or more IT

[2]The TAPE method has been published in the following book chapter: Müller, M. and Ostern, N., Rodriguez Garzon., S. and Küpper, A. (2021) Engineering Trust-aware Decentralized Applications with Distributed Ledgers. Trust Models for Next Generation Blockchain Ecosystems. EAI/Springer.

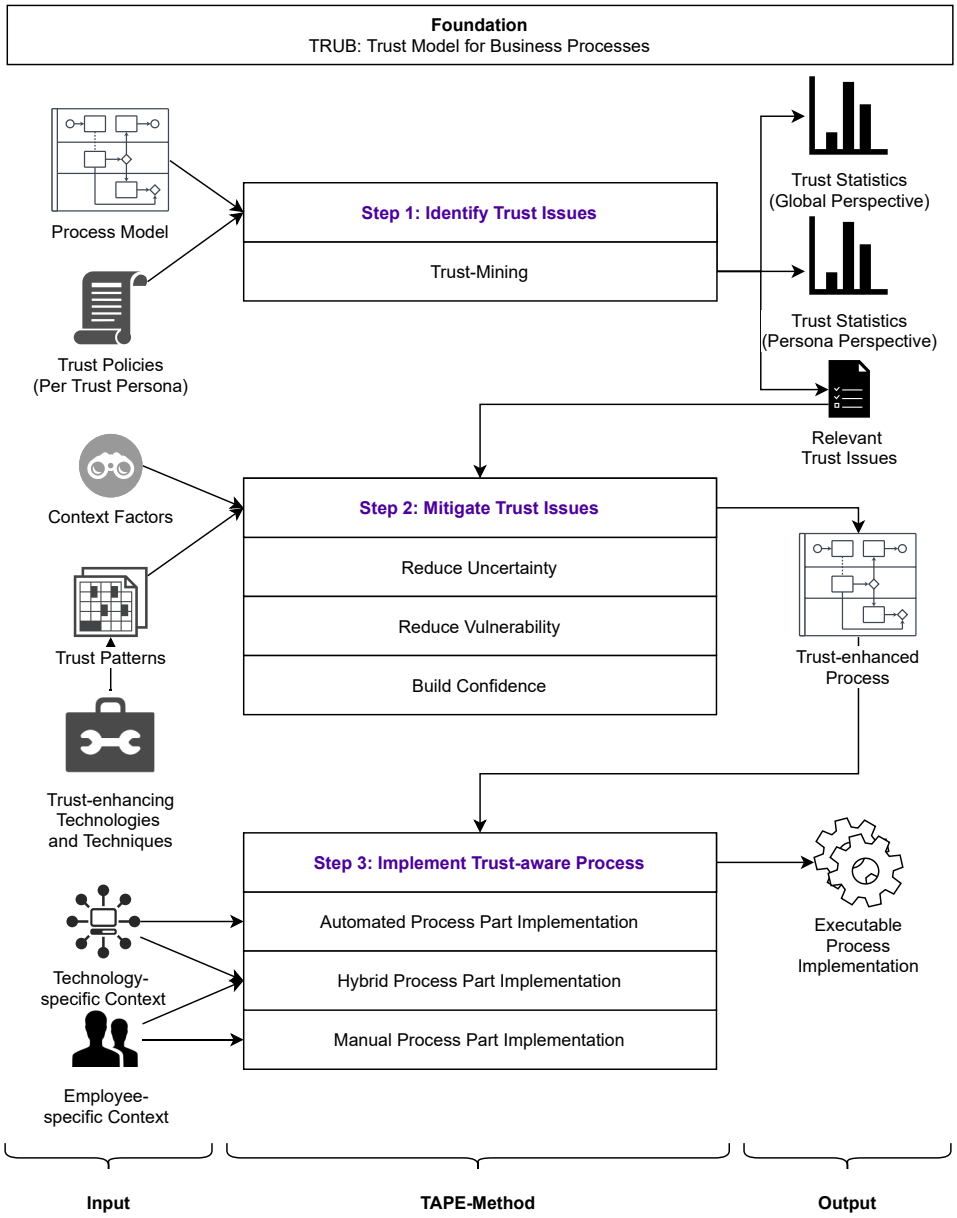

Figure 3.3: The three-step-approach utilized in the TAPE method.

Systems, hybrid parts where humans need to interact with a system and manual process parts alike. Section 3.7 elaborates on Step 3 in detail.

3.4 A Formal Model for Trust in Collaborative Business Processes (TRUB)

This section presents a formal model for trust in business processes (TRUB). It constitutes the base for the three steps of the TAPE method. TRUB aims to formalize the highly subjective notion of trust in a way where uncertainties can be attributed to certain parts of a process[3].

[3]The TRUB model has been published in the following journal article: M. Müller, S. R. Garzon, M. Rosemann and A. Küpper, "Towards Trust-Aware Collaborative Business Processes: An Approach to

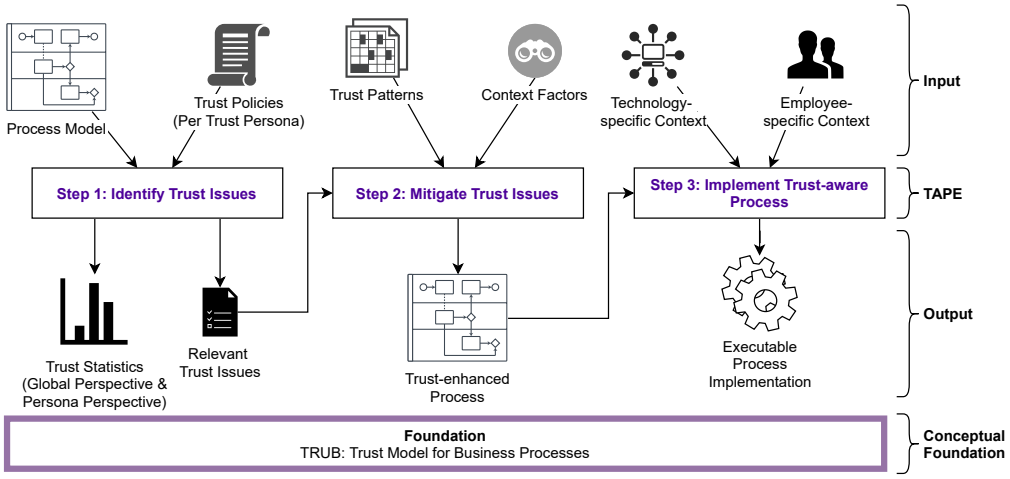

Figure 3.4: TRUB in the context of TAPE.

3.4.1 Related Trust Models

TRUB utilizes concepts and methods already proposed in different domains such as trust management or the web of trust and adapts them to the domain of BPM. The following sections give an overview of the origin of utilized concepts.

The Web of Trust is a concept used in systems to establish the authenticity of a public key of an entity by having other already known entities certifying its authenticity [145]. In such a system, trust is created through a decentralized network of certification statements. This approach stands in contrast to public key infrastructures (PKI) where a single centralized party certifies all other parties [146]. Different entities certifying others can be represented in a graph. In such a graph the edges represent the certification and, thus, the trust relationship that creates the figurative "web" of trust. Different other models for decentralized trust management exist. They are all focused on the technical aspects of security that use some of the concepts of the Web of Trust [147]. While the application of the Web of Trust as a decentralized trust model and concepts on top of it is targeted for the specific use case of online identity, the trust model used to assess the trustworthiness of certain actors can be generalized and applied to the business processes. Khare and Rifkin [148] propose to think about *principals* (who to trust) and policies towards them following different *principles* (whom to trust for what). This structure will be reused partially for TRUB.

A similar concept of describing whom to trust for what was also proposed in Grandison and Sloman's SULTAN model for specifying and analyzing trust in internet applications [144]. The concept defines a *trust construct* that describes a *trustor* trusting or distrusting a *trustee* in the context of a specified *action set* to a certain *trust or distrust level* under *constraints*. This model can describe situations like Willy (trustor) trusts the bank (trustee) for the correct opening of a new bank account (action set) to a trust level of 50% during normal business operation (constraint). The same concept also provides a *recommendation construct* in which a *recommender* recommends or does not recommend a *recomendee* with a confidence level to perform an *action set* to a *confidence level* under certain *constraints*.

Identify Uncertainty," in IEEE Internet Computing, vol. 24, no. 6, pp. 17-25, 1 Nov.-Dec. 2020, doi: 10.1109/MIC.2020.3023180.

For example, Sally (recommender) recommends Whole Foods (recommendee) for purchasing vegan tomato soup (action set) very highly (confidence level) during the summer months (constraint).

The utilization of recommendation as a form of leveraging relationships to assess trustworthiness is one form of trust management. Another form of trust management is the utilization of reputation. Reputation is the *perception* a party creates through past actions about its intentions and norms [33]. Conceptionally, reputation statements consist of a reputation *source* that makes a reputation *claim* towards a certain reputation *target*. Claims can be quantitative (e.g., normalized scalar values or rank values) or qualitative (e.g., free form text or media uploads). The source and target of reputation statements can be any entities or reputation statements themselves. Reputation systems store reputation statements, aggregate them, and transform them into a meaningful form to the user.

The exploitation of trust relationships using reputation and recommendations has also been the foundation of network-related approaches that offer a more in-depth analysis of the relationships themselves. Jøsang et al. [149] propose a method to analyze trust networks that uses logic expressions to derive relationships. They propose to introduce transitive trust relationships. For example, A trusts B and B trusts C, then A transitively trusts C. With the trust relationships, a graph can be built that shows all direct and indirect relationships and dependencies. The concept of trust dependencies will also be used in a different form in TRUB. Network-centric trust models as proposed by Jøsang et al. [149, 150] have seen many different adoptions and have been applied to various application domains ranging from traditional social networks [151] to human-robot interactions [152].

3.4.2 Requirements for a Process-centric Trust Model

TRUB builds upon different concepts of the previously discussed approaches in the context of a business process. The trust model combines trust-centered literature and BPM elements into a new model to describe and analyze trust in collaborative business processes. The focus on trust in a process, its components, and collaborators executing it leads to a more fine-granular model than currently existing approaches. This model enables its users to attribute different uncertainties and trust relationships to separate parts and interactions in a process. Therefore, the focus on a process implies different requirements to the trust model. The requirements are derived from the general goal of providing a business process model as an input to the trust analysis. The next paragraphs provide a short overview of the requirements for TRUB.

No Bootstrapping Models utilized in traditional trust management systems, such as reputation systems or recommender systems, require an initial set of information through which the trustworthiness can be assessed. This initial dataset's sourcing is often called bootstrapping and a common challenge to trust management systems [153]. In many use cases of collaborative business processes, it is not feasible to create trust management systems. For example,

in business-to-business interactions maintaining an online reputation system for the execution of different business interactions is often prohibited by privacy clauses in the terms of service. For centralized multi-sided platforms, the platform might maintain a trust management system towards its users but might not be interested in reputation statements towards the platform itself. TRUB must be an ad-hoc method to address these shortcomings. TRUB should be applicable to any process without any collection of past experiences or recommendations.

Different Subjects of Trust A trust model for collaborative business processes should be able to investigate different subjects of trust. In heterogeneous business processes, a single process participant might have different trust preferences towards the same collaborator. For example, an e-commerce business might trust an airfreight carrier to create the right shipping labels through its online system. The business might not trust its collaborators that the system will be available for more than 80% of the time needed due to bad experiences with the system in the past. Moreover, different trust concerns for the same actions might cause varying trust preferences. Furthermore, different parts of a process might be regarded differently regarding trust. For example, an e-commerce business might trust an international carrier for flying a parcel to the right country and loading it in a post truck. Yet, the e-commerce business operators might not trust the carrier to repeatedly come to the final recipient's door in case she cannot be found on the first attempt. This is a common problem in last-mile delivery actions that illustrates that a trustor might have different trust preferences for different activities.

Process Model Independence The final requirement for a process-centric trust model is the possibility of adapting it to standard business modeling languages. The trust model should be independent of a specific process modeling language. Adapting standard business process notations allows domain-independent benefits and automated analysis of the model. Automation is a requirement to improve the quality of the trust analysis since human analysis is error-prone.

3.4.3 TRUB Overview

The TRUB trust model utilizes business process models as a base. TRUB is independent of specific instances of business process models and can be applied to different modeling languages (e.g., BPMN, YAWL, or Petri Nets) that consist of the following elements:

- **Basic Nodes** Trust Mining-compatible modeling languages need capabilities to express activities, events, and gateways. These classes of elements are the basic building blocks of many process model [37].
- **Swimlanes** In collaborative business processes, workflow elements are always associated with one organization. Separate organizations are responsible for different parts of the overall process. Swimlanes are commonly used in different process modeling languages to express organizational borders and responsibilities.

- **Data Modeling** Data input and output elements are required to indicate data flow. In conjunction with swimlanes, they also express data origin and processing responsibility.
- **Connections** Any process modeling language compatible with Trust Mining needs to connect elements within an organization (sequence flow) or across different organizations (message flow). These connections are the elements that order process elements to a workflow.

The remainder of this work utilizes an independent meta-modeling language for collaborative business processes (MCBPM) that is created based on these requirements. Popular "real" process modeling languages can be translated to MCBPM. The syntax is similar to a simplified version of BPMN 2.0. Petri nets can also be translated to MCBPM by using different extensions [154]. The meta-model is inspired by similar formal notations as introduced by van der Aalst [40].

$$p = (N, E, L, \lambda) \tag{3.1}$$

A process p consists of a set of nodes N, a set of edges E, a set of lanes L, and a function λ assigning nodes and edges to lanes. Nodes symbolize different activities, gateways, or events by using an *element type* description $et \in ET$. These types are assigned to nodes with the function τ. By default, MCBPM uses activities, splits, joins, data, and events as element type descriptors. It is also possible to model all node types from the extensive set included in the BPMN 2.0 specification or the limited number of Petri Nets elements.

$$\begin{aligned} N =& \{n_0, ..., n_k\} \\ ET =& \{\text{activity, start, end,} \\ & \text{intermediate, split, join,} \\ & \text{data-in, data-out}\} \\ \tau :& N \rightarrow ET \end{aligned} \tag{3.2}$$

Edges connect the different nodes in the process. They either express control flows or message flows. Control flows indicate a workflow that is executed within one organization. Messages represent interactions between actors from different organizations. In general, edges are a tuple of nodes. The function η illustrates whether an edge represents control or message flow.

$$\begin{aligned} E &= \{e_0, ..., e_k\} \\ e &= (n_i, n_j), n_i, n_j \in N \\ \eta &: E \rightarrow \{\text{control}, \text{message}\} \end{aligned} \tag{3.3}$$

Swimlanes L model different organizations being associated with different elements of the process. MCBPM does not distinguish further between pools (different organizations) and intra-organizational lanes (within an organization)

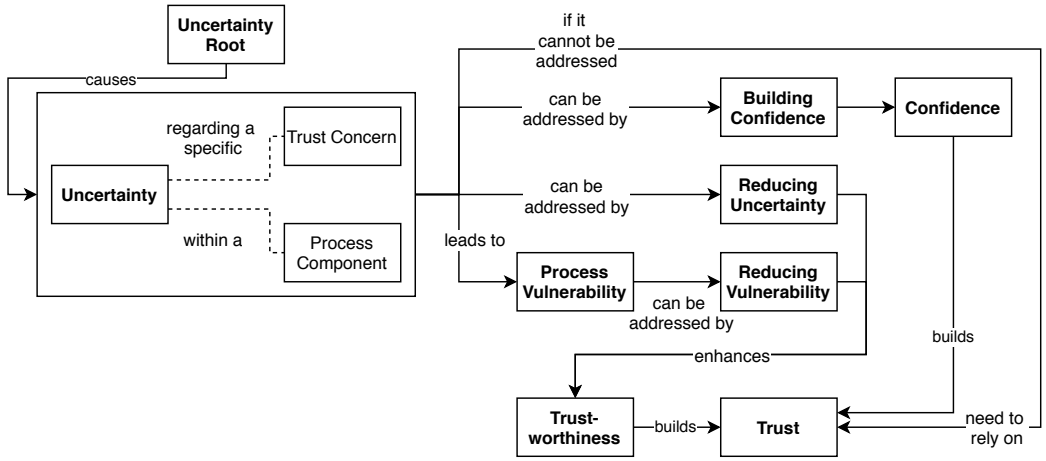

Figure 3.5: Relationship of uncertainty, vulnerability and trust in the context of business processes. Extended version based on [9]. Reprint with permission from [155].

as BPMN does. This limitation is based on the assumption that every organization trusts itself. Thus, every (swim-)lane is atomic and represents a whole organization. Formally, process nodes get assigned to a (swim-)lane $l \in L$ with the function λ.

$$\begin{aligned} L &= \{l_0, ..., l_n\} \\ \lambda &: N \rightarrow L \end{aligned} \tag{3.4}$$

With MCBPM as a generic business process model foundation, Figure 3.5 illustrates the architecture of TRUB. The model describes that a certain *root of uncertainty* causes *uncertainty* that leads to a specific *trust concern*. That uncertainty can be located within a *process component* in the process model. The presence of uncertainty leads to potential *vulnerabilities* of the process. A vulnerability describes the impact when a part of the process or the whole process is not executed as desired. Trust in the process can be enhanced *reducing uncertainty*, *reducing vulnerability* or *building confidence*. The reduction of uncertainty aims to lower the probability *that* something undesired happens. On the other hand, reducing vulnerability aims to mitigate the impacts *when* something undesired occurs in the process. Reducing uncertainties and vulnerability are direct approaches to increase the trustworthiness of the process. In contrast, *building confidence* is an indirect approach that introduces additional sources of trust in the process.

To illustrate the principles of TRUB better, Figure 3.6 shows an example business process model of an apartment leasing process through a multi-sided platform comparable to AirBnB. The process model uses the BPMN 2.0 syntax for illustration, while in general, any process modeling language that is compatible with MCBPM can be utilized. The process starts with a tourist who has made a decision to book a certain apartment for vacation in a new city. The tourist makes a deposit to the multi-sided platform, which handles the tourist's money as an escrow. The platform contacts the apartment owner with the booking details. Then all participants wait for the booked period to arrive. On the first day of the

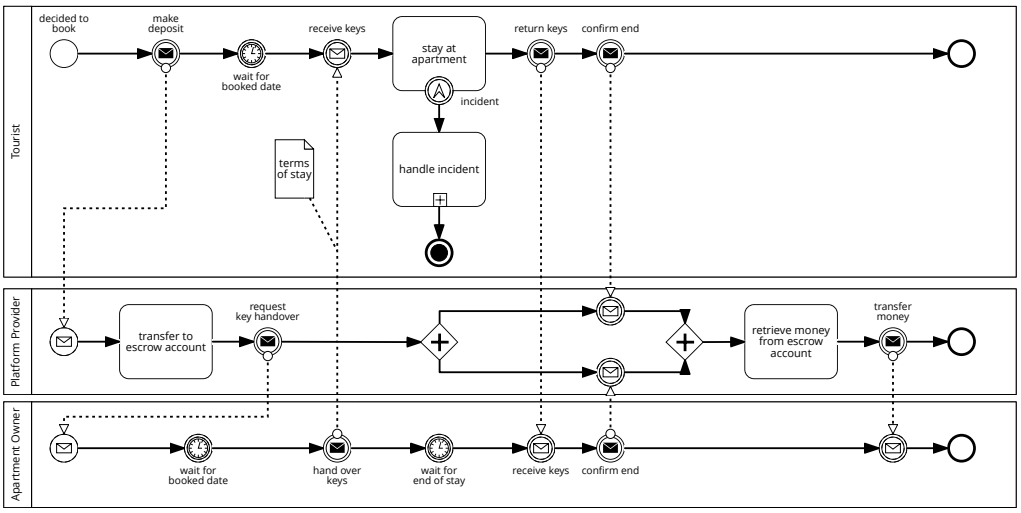

Figure 3.6: The figure shows an apartment leasing process with a multi-sided platform. The model uses BPMN 2.0 with a small modification. In the diagram, the terminate event terminates all tokens regardless of their association with a lane. This modification is introduced for simplicity. The syntax is fully compatible with the MCBPM meta model.

rental, the apartment owner hands over the keys to the apartment to the tourist. In this handover process, the apartment owner also gives the tourist a piece of paper with the terms of stay printed onto it. The terms of stay define how the tourist has to behave, e.g., not yelling on the balcony at 3 am, and what to do when something unexpected happens that damages the apartment. With these instructions, the tourist starts the vacation period. If something unexpected happens, for instance, if the tourist sets the kitchen accidentally on fire while cooking, it is required to create an incident and handle it with the platform and the apartment owner. Incident handling is an external subprocess that is not modeled in this process. If the tourists' stay is over without any incident, the tourists return the apartment keys to the apartment owner. Afterward, both are required to signal the platform that the stay has ended. With this information from both parties, the platform provider can release the money from the escrow account to the apartment owner and finish the process.

Concerning the apartment rental example, TRUB can describe many uncertainties. For example, the tourist (root of uncertainty) causes uncertainty regarding the integrity (trust concern) of the "stay at apartment" activity (process element). This means that there is the possibility that the tourist does something undesired during the execution of this activity. It might refer to incidents like setting the kitchen on fire. Another possible uncertainty might be caused by the platform provider (uncertainty root) about the confidentiality (trust concern) of the money transfer to the escrow account (process element). If this activity's confidentiality is not assured, it might be possible that other platforms can analyze the business figures of the platform and its users. This might be undesired by the process participants.

With this general overview of the TRUB elements, the next sections provide an

in-depth discussion on the uncertainty possibility description part of the TRUB model.

3.4.4 Uncertainty Root

In TRUB, an uncertainty root describes *who or what* can potentially cause an uncertainty. This description of responsibility to an uncertainty can be formally defined through an uncertainty root set R.

$$R = \{r_0, ..., r_n\} \tag{3.5}$$

TRUB explicitly does not define fixed uncertainty roots. Moreover, TRUB lets the user define the explicit uncertainty roots. One possible uncertainty root set R_{org} might use the different organizations, i.e., the lanes in the process model, to attribute uncertainties to.

$$R_{org} = L \tag{3.6}$$

With that schema, all uncertainties related to the "stay at apartment" activity in the apartment rental example can be attributed to the tourist as uncertainty root, while the platform provider is the uncertainty root of the "transfer to escrow account" activity.

If the user wants to use TRUB to attribute uncertainties to an "involved subject", R_{subj} can be defined as a set consisting of software, humans, or an organization.

$$R_{subj} = \{\text{software}, \text{human}, \text{organization}\} \tag{3.7}$$

Utilizing R_{subj} as the uncertainty root set to apply TRUB to the apartment example, all uncertainties for "stay at apartment" activity can be attributed to a human (i.e., the tourist), while the transfer of the money to an escrow account can be attributed to the automated software service of the platform that interacts with the escrow.

The possible set of R sets is not limited to the two discussed possibilities. TRUB leaves the R set open for definition as a model parameter. Yet, it is fixed for the whole process under investigation with TRUB.

3.4.5 Trust Concerns

Similar to uncertainty roots, the TRUB model uses a predefined set of trust concerns as a configuration parameter, but leaves it up to the user which trust concerns to use. In contrast to models where trust concerns are completely open, as proposed by Grandison and Sloman [144], this approach enables completeness

within the set of defined trust concerns. This implies a closed-world assumption. Formally, TRUB defines these trust concerns as a set TC:

$$TC = \{tc_0, ..., tc_n\} \tag{3.8}$$

In the context of collaborative business processes, no consensus on a set of trust concerns has been established yet. To identify from which fields relevant trust concerns for business processes originate, it is important to discuss the fundamental roots of BPM. BPM is a subfield of information systems engineering. In business processes, different process parts are executed by different organizations. These organizations share and store information. Thus, the field of *information security* is a relevant field to derive trust concerns for BPM. Since large-scale systems are part of a process in many cases, *system-related properties* may also be used to derive different trust concerns. Especially in collaborative business processes, where different organizations work on a shared process, the separate subprocesses can be seen as a service provided to the other collaborators. Thus, *quality of service attributes* and their fulfillment may also be a suitable set of trust concerns for collaborative business processes.

The following paragraphs provide an overview of trust-related properties in the mentioned fields. Afterward, a reference set of trust concern is compiled. It is a combination of trust concerns of different origins and will be utilized throughout the remainder of this work. The origin sets are meant to be a reference suggestion for trust concerns. In general, the trust model in this section and its methods in the following sections can use any set of trust concerns. Hence, the suggested reference set of trust concern is by no means a claim to completeness.

In the field of information security, the CIA-triad (as defined in ISO 27000 [156]) consisting of confidentiality, integrity, and availability is a commonly used description of information security properties.

$$TC_{CIA} = \{\text{confidentiality}, \text{integrity}, \text{availability}\} \tag{3.9}$$

The extended ISO 27000 definition for trust concerns regarding information security management systems includes in addition to the CIA-triad also authenticity, accountability, non-repudiation and reliability.

$$TC_{ISOext} = TC_{CIA} \cup \{\text{authenticity}, \text{accountability}, \text{non-repudiation}, \text{reliability}\} \tag{3.10}$$

Ross and McEvilley describe an approach to engineering trustworthy secure systems for NIST 800-160 [157]. They introduce safety, security, reliability, dependability, performance, resilience, and survivability as example requirements for a trustworthy system. In their main concept on how to engineer trustworthy

secure systems, they mention that the relevant, trustworthy requirements depend on the focus of the systems engineer and the objective of the system as a whole.

$$TC_{NIST} = \{\text{safety}, \text{security}, \text{reliability}, \text{dependability}, \text{performance}, \text{resilience}, \text{survivability}\} \tag{3.11}$$

In the field of quality of service, especially in information systems, the Object Management Group (OMG) defines performance, security, integrity, coherence, latency, efficiency, demand and reliability as quality of service attributes [158].

$$TC_{QoS} = \{\text{performance}, \text{security}, \text{integrity}, \text{latency}, \text{efficiency}, \text{demand}, \text{reliability}\} \tag{3.12}$$

Mohammadi et al. describe in their work trustworthiness attributes of software systems in the context of "Socio-Technical Systems" [159]. They obtained these attributes by a survey of 138 papers that deal with trustworthiness in software systems. Semantically, these attributes focus on software quality and can be considered trust concerns. They concluded a set of reference trustworthiness attributes and quantified in how many of the surveyed papers of software trustworthiness attributes they occurred.

- **Security (63%)** including accountability, auditability, traceability, confidentiatlity, integrity, and safety
- **Dependability (44%)** including accurracy, availability, fault tolerance, reliability, robustness, scalability and maintainability
- **Usability (26%)** including satisfaction, learnability, effectiveness and efficiency of use
- **Privacy (18%)**
- **Correctness (17%)**
- **Data-related Quality (17%)** including data integrity, data reliability, data timeliness, and data validity
- **Compatibility (11%)** including openness, and reusability
- **Performance (11%)** inclcuding transaction time, throughput, and response time
- **Configuration-related Quality (10%)** including stability and completeness
- **Cost (6%)**
- **Compliance (4%)**
- **Complexity (3%)**

	TC_{CIA}	TC_{ISOext}	TC_{NIST}	TC_{OMG}	TC_{PR}
Confidentiality	●	●	○	○	●
Integrity	●	●	○	●	●
Availability	●	●	○	○	●
Security	○	○	●	●	○
Authenticity	○	●	○	○	○
Accountability	○	●	○	○	○
Non-repudiation	○	●	○	○	●
Reliability	○	●	●	●	○
Performance	○	○	●	●	●
Safety	○	○	●	○	○
Dependability	○	○	●	○	○
Resilience	○	○	●	○	●
Survivability	○	○	●	○	○
Latency	○	○	○	●	○
Efficiency	○	○	○	●	○
Demand	○	○	○	●	○

Table 3.2: Comparison of different attributes that may be used for trust concerns.

The TRUB trust model can use any set of trust concerns as configuration parameters. If a process engineer only wants to assess trust concerns related to information security properties, the engineer can use TC_{CIA} or TC_{ISOext}. The remainder of this work will use a set of exemplary trust concerns that utilize trust concerns from the different examples discussed before as process-related trust concerns called TC_{PR} as illustrated in Table 3.2.

The TC_{PR} set includes integrity, confidentiality, availability, and non-repudiation, performance, and resilience. Thus, it combines the different aspects from information security [160, 161, 156], trustworthy systems [157], and quality of service [158]. The argument why exactly these trust concerns are utilized is structured as follows: Information systems handle data that is transferred between and processed by different organizations. Hence, the widely established CIA properties are used as trust concerns. Integrity and availability can semantically also be applied to other aspects of a process than purely for information. For instance, a trust concern regarding the integrity of the execution of an activity by a collaborator can semantically also describe the uncertainty concerning the correctness of the execution itself. With the goal of utilizing trust concerns that are meaningful in the context of different process concerns, this approach helps to keep the set of trust concerns minimal, while semantically covering different aspects. As the last trust concern derived from the security domain, non-repudiation is also included in TC_{PR}. This property is included because especially in the coordination between different organizations, it is important that certain states have been reached and cannot be reverted to ensure the proper execution of the collaboration. From the QoS domain, TC_{PR} includes performance and resilience. Performance can be seen as a parent of latency, efficiency, and other non-functional service properties. Resilience partially also covers

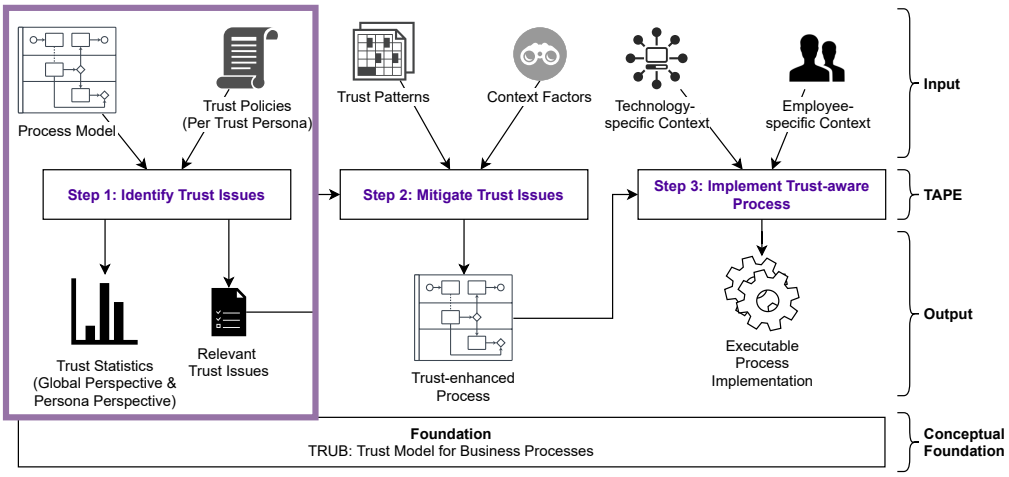

Figure 3.7: Trust Mining in the context of TAPE.

surviveable and dependability.

TC_{PR} is throughout the remainder of this work utilized as a mere example set of trust concerns without any claim to completeness or universality. Like in the cause of the uncertainty root, TRUB leaves trust concern up to the process engineer to define as a configuration parameter of the model.

3.4.6 Process Elements

The TRUB trust model can use any set of process elements for the utilized process model, as long as they are compatible with the described MCBPM meta-language. Since the currently used standard modelling languages like BPMN 2.0 or Petri Nets can be translated to MCBPM, the TRUB trust model is also capable of adapting them.

3.5 Identify Trust Issues in Business Processes: Trust Mining

The first step of the TAPE method is concerned with the identification of trust issues in business processes. Therefore, the following sections introduce the concept of Trust Mining. Trust Mining is a novel method for the analysis of trust issues using business process models as a tool for model-driven software engineering[4].

3.5.1 Requirements for Trust Issue Identification

Trust Mining aims to be as general as possible while still providing tangible insights into the trust-related concepts in a process. Therefore, Trust Mining needs to accommodate the following requirements:

[4]Trust Mining has been published in the following journal article: M. Müller, N. Ostern, D. Koljada, K. Grunert, M. Rosemann and A. Küpper, "Trust Mining: Analyzing Trust in Collaborative Business Processes," in IEEE Access, vol. 9, pp. 65044-65065, 2021, doi: 10.1109/ACCESS.2021.3075568.

Automation Previous work of trust analysis often followed an open-world assumption. For example, Grandison and Sloman assess trustworthiness by surveying different stakeholders [20]. Without any limitations on trust concerns, identifying trust is challenging. Furthermore, an open-world assumption regarding the trust concerns, process elements, and requirements does not allow for comparability. Without any restrictions on the trust-related aspects, the identification process is not automatable and needs to be executed mostly manually. Manual approaches lead to human errors. Thus, Trust Mining aims to be as automated as possible.

Semantic Generalization For describing trust, the different semantics in business processes influence the overall analysis. Business process models introduce meta-semantics to a process. An activity represents a unit of work. Yet, the work of two activities might be fundamentally different. For example, one activity "log in to website" and another activity "handover parcel" might have different semantics for the same trust concerns. Confidentiality in the first activity might be concerned with the security of the user's password. In the second activity, the confidentiality trust concern can be related to the handover not being visible to competitors or criminals that might want to damage the physical object. Trust Mining needs to be semantically general while still being semantically compatible with a large number of processes.

Process Modeling Language Independence Trust Mining should not be designed exclusively for one process modeling language. It should enable adaption to different standards such as BPMN or Petri nets.

Measurements Trust is an abstract social concept. Yet, analyzing the impact of trust issue mitigations requires measuring the difference between the trust issues in the initial process and the improved process. Thus, establishing metrics for trust-related concepts in the contexts of a business process is essential. Trust Mining needs to define different measurements for different goals.

3.5.2 Trust Mining's Four Steps

This section introduces Trust Mining as a method to understand uncertainty and its impacts within a business process. Trust Mining takes place during design time within the BPM lifecycle. The approach is a base to implement business processes complying with the trust tolerance profiles of different process stakeholders. These stakeholders act as *trust personas* and define their trust policies.

Conceptually, Trust Mining builds upon the TRUB model as introduced in Section 3.4. The concept consists of four steps, as illustrated in Figure 3.8. The first three steps embody a global trust analysis, while the final step assesses the relevancy of certain trust issues from the perspective of specific trust personas. Trust Mining takes a business process model and a set of trust policies. Trust policies are associated with trust personas and represent trust tolerance profiles.

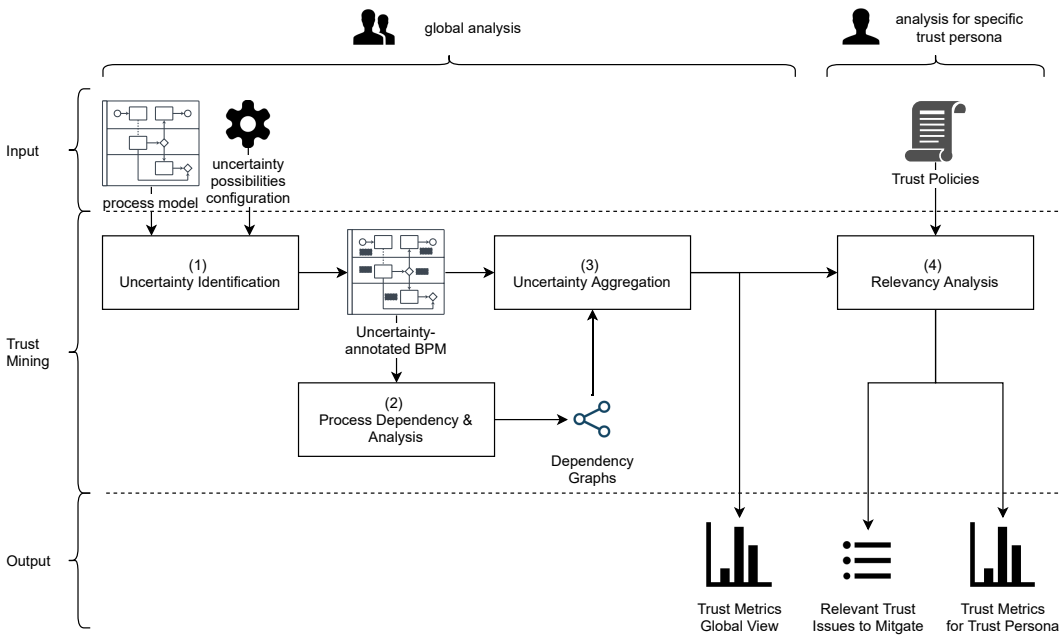

Figure 3.8: The four step process of Trust Mining: Uncertainty Identification, Process Dependency Analysis, Uncertainty Aggregation and Relevancy Analysis.

In addition, Trust Mining requires a definition of uncertainty possibilities as a case-independent configuration parameter.

In Step 1, Trust Mining identifies uncertainties in a business process. Therefore, the method uses a process model and a list of uncertainty possibilities. The configuration defines the uncertainties regarding specific process components. The outcome of this step is an annotated business process model. The annotation illustrates which uncertainties could potentially lead to trust issues in specific parts of the process. Hence, Step 1 adds a *trust layer* to the process model. The trust layer is employed as the input to the later steps.

Step 2 in Trust Mining comprises of a process dependency analysis that builds upon the uncertainty-annotated process model. In this step, Trust Mining analyzes process dependencies between different process components as defined in the model. This analysis uncovers process-related functional and non-functional trust dependencies. The outcome of Step 2 are graphs for message and data dependency. They illustrate the relationships between the different organizations regarding uncertainties in message and data exchanges.

Step 3 uses the uncertainty-annotated process model from Step 1 and the relationship graphs from Step 2 as an input. In this step, different trust metrics illustrate the uncertainties in a process to get a better understanding of trust issues. The calculations are process-wide or an aggregation depending on the subprocess that the involved organizations are responsible for. The aggregation shows uncertainty distribution and other relevant trust statistics to quantify uncertainties.

The relevance analysis in Step 4 observes how trust issues affect trust personas (process stakeholders and collaborators). Therefore, trust policies express the trust tolerance profiles of different trust personas. Relevancy analysis marks

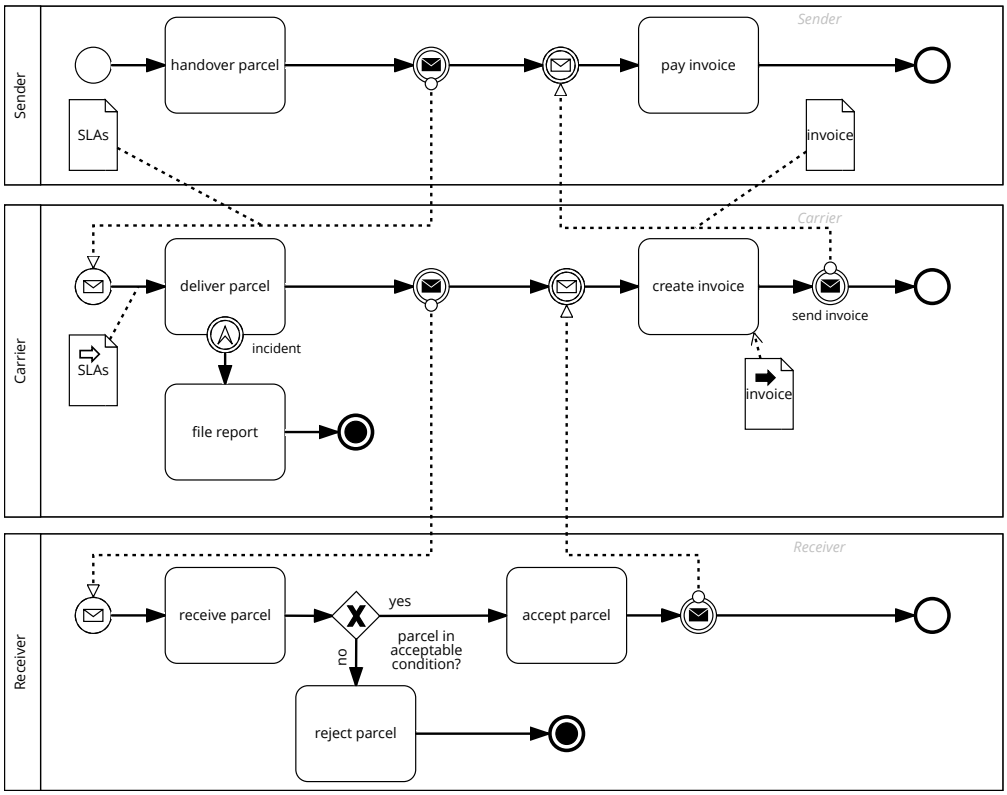

Figure 3.9: A process of supply chain of dangerous goods. The diagram adheres to the BPMN 2.0 notation with a modification. For simplicity, this example assumes that the process termination element terminates all tokens in all lanes instead of only the current lane as specified in the BPMN 2.0 standard.

the last step in Trust Mining. It delivers a list of trust issues as an output for each trust persona. These outputs are used as a base for purely analytical purposes (e.g., convincing someone of the trustworthiness of a process) or for the trust-aware implementation of the business process by mitigating trust issues.

Running Example The following in-depth introduction of Trust Mining illustrates the principle referencing the running example of the supply chain of dangerous goods in Figure 3.17. The goal of the process is to deliver a parcel with dangerous goods from a sender to a receiver. The transport of dangerous goods, such as firework rockets or explosives in general, has unique requirements. Parcels need to be kept in an anti-static environment. Otherwise, sparks could ignite the fireworks during their delivery. In the process, the sender packs the objects into a package. The sender defines service level agreements (SLAs) in an SLA document. These agreements state how the parcel needs to be handled by the carrier. In this case, the SLAs specify detailed instructions regarding the required anti-static environment. The carrier needs to adhere to these SLAs. Once the carrier's delivery agent reached the sender for the parcel pickup, they interact with each other. The sender hands the SLA document and the parcel over to the carrier. The carrier places the parcel in the truck and delivers it to the receiver.

In case an incident occurs, the carrier needs to create a report and terminate the delivery process. Incidents are all events that violate the SLAs, for example, an exploding firework rocket. If the carrier reaches the receiver without an incident, the carrier hands the parcel over to the receiver. The receiver evaluates the parcel's conditions. In case the receiver detects that an SLA violation occurred, the receiver rejects the parcel. Otherwise, the receiver accepts. After the delivery is finished, the carrier creates an invoice. The price of the delivery depends on the SLAs, the traveled time, and the distance. The carrier sends the invoice to the sender. The sender pays the invoice. This represents the end of the process. From the view of the sender trust persona, the initial process has trust issues. TAPE's later steps aim to mitigate them.

3.5.3 Process Model for Trust Mining

Trust Mining is independent of any business process modeling language. The method can be applied to any modeling language that has the four main capabilities of modeling basic nodes, swimlanes, data, and connections. The MCBPM meta-model introduced in Section 3.4.3 provides all required capabilities. Furthermore, Trust Mining uses TRUB with MCBPM as its core. Concrete modeling languages can be adapted to the meta-model.

The running example for the delivery of dangerous goods, as illustrated in Figure 3.9, can be translated to the meta-model as follows. The process p^{ex} consists of nodes, edges, lanes, and a function to assign the elements to lanes. Nodes include all activities, events, gateways, and data objects defined in the process model. This includes, for example, the start event n_{start}, the prepare parcel activity $n_{\text{prepare parcel}}$, or the SLA data output $n_{\text{SLA out}}$.

$$
\begin{aligned}
p^{ex} &= (N^{ex}, E^{ex}, L^{ex}, \lambda^{ex}) \\
N^{ex} &= \{n_{\text{start}}, n_{\text{prepare parcel}}, \\
&\quad\ n_{\text{SLA Out}}, \ldots, n_{\text{end}}\} \\
\tau(n_{\text{start}}) &= \text{start} \\
\tau(n_{\text{prepare parcel}}) &= \text{activity} \\
\tau(n_{\text{SLA out}}) &= \text{data-out} \\
&\ldots \\
L^{ex} &= \{\text{sender}, \text{carrier}, \text{receiver}\} \\
\lambda^{ex}(n_{\text{prepare parcel}}) &= \text{sender} \\
&\ldots
\end{aligned}
\tag{3.13}
$$

Translating the complete running example in Figure 3.9 leads to 30 nodes (all tasks, data objects, events, and the gateway). The nodes are connected with 31 edges (20 control flow, 4 message flows). The inter-organizational process flow is divided into three lanes (sender, carrier, receiver).

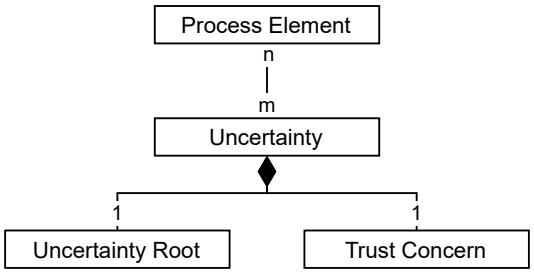

Figure 3.10: Schematic model of the connection of process elements, uncertainties, trust concerns and uncertainty roots.

3.5.4 Trust Model and Configuration Parameters

Fundamentally, Trust Mining uses a process model and a trust model as its two core concepts. The MCBPM modeling language in conjunction with the TRUB model as a trust layer are flexible components used in Trust Mining. The trust model expresses that *uncertainty roots* cause *uncertainty* regarding a specific *trust concern*. This uncertainty can be associated with a *process element* in a business process model. In general, a process element can have many uncertainties. Different uncertainties are relevant for separate process elements. This creates a many-to-many relationship as illustrated in Figure 3.10.

Trust Mining needs to semantically understand which uncertainties are possible and of relevance for certain process elements. Therefore, the Trust Mining method defines an *uncertainty possibility list (UPL)*. This list states, for example, that in an activity (process element) that is executed by an employee as a manual task, this employee (uncertainty root) may cause uncertainty regarding the correct execution (i.e., integrity, the trust concern) of the activity. The UPL defines such relationships independent of the detailed semantics of the process elements. Thus, this holds for *all instances of activities*.

Formally, the UPL consists of n uncertainty possibilities up. Each up is defined as triplet of an element type et, a trust concern tc, and an uncertainty root r.

$$\begin{aligned} UPL &= \{up_1, ..up_n\} \\ up &= (et, tc, r) \\ et &\in ET \\ TC &= \{t_1, ..., t_n\} \\ tc &\in TC \\ R &= \{r_1, ..., r_n\} \\ r &\in R \end{aligned} \tag{3.14}$$

In general, Trust Mining may use any set of trust concerns and uncertainty roots. Thus, they act as *configuration parameters* of the utilized trust model. They define how the UPL can be created. This work utilizes reference sets for the configuration parameters. For uncertainty roots, the following sections employ software, human resources, and organizations as uncertainty roots, according

Algorithm 1: createUncertaintyPossibilityList

```
Input: ET, R, TC
Result: UPL
UPL := {} ;
foreach et ∈ ET do
    foreach tc ∈ TC do
        foreach r ∈ R do
            if rootCanCauseUncertaintyAtElement(r, tc, et) then
                if uncertaintyIsSemanticValid(r, tc, et) then
                    up_new := (r, tc, et) ;
                    UPL := UPL ∪ up_new ;
                end
            end
        end
    end
end
return UPL
```

to R_{subj} as defined in Section 3.4.4. The reference set of trust concerns comprises integrity, availability, confidentiality, non-repudiation, performance, and resilience, following TC_{PR} as defined in Section 3.4.5. However, Trust Mining is not limited to these reference sets. Process engineers and analysts using Trust Mining can change these configuration parameters as desired.

Creating the UPL Creating the UPL follows the pseudocode in Algorithm 1. The algorithm iterates over all possible combinations of uncertainty roots r, trust concern tc and element type et. For each combination, the creator of the UPL needs to assess two questions to determine whether or not to include the uncertainty possibility in the UPL.

1. Is it possible that an *uncertainty root* r may cause uncertainty regarding the *trust concern* tc in a *process element* et? This question assesses the general circumstances independent of any concrete process under which uncertainties may exist. For example (A), is it possible that a human resource may cause uncertainty regarding the confidentiality of a manual task? For example (B), is it possible that an employee may cause uncertainty regarding the integrity of a message transfer?

2. If the creator of the UPL determines that a root r can in theory cause uncertainty regarding a trust concern tc at an element et, the next question applies. Is the uncertainty possibility semantically valid? In Example A, it is semantically valid that a human resource might, in some instance of a manual task, cause an uncertainty regarding its confidentiality. For an employee of a bookkeeping firm who fills out tax sheets manually, it is possible that the employee could leak that information to unauthorized parties. In Example B, the question is semantically not valid. In the general classes of processes that a process engineer assesses, human resources are never responsible for the message exchange. Instead, the organization globally coordinates the communication with other organizations on a higher level

of responsibility. These assumptions rely on the underlying principles that a process engineer assumes.

Only if both questions are valid, the triplet of r, tc, and et gets added to the UPL. This approach illustrates how semantics are injected into Trust Mining. The validity of uncertainty possibilities depends on the subjective judgment of the UPL's creator. In that way, UPLs might differ depending on the process engineers' and analysts' assumptions.

Reference UPL This paragraph defines a reference UPL that will be utilized throughout the remainder of this chapter. The list is depicted with an illustrative question for each uncertainty possibility in Tables 3.3 and 3.4.

The roots of activity-related uncertainty are, in most cases, human resources or software systems that execute the activity. Uncertainty regarding the integrity trust concern is focused on whether an activity is executed correctly. Confidentiality refers to privacy requirements of the execution. For instance, some activities in a process may be specific routines that an organization wants to keep private for competitive reasons. Availability describes the uncertainty that all required (software or human) resources are available at the time of execution. If the driver in the running example is not available to drive the dangerous parcel to its destination, it might come to problems in the process. Non-repudiation is concerned with the undeniability of an organization that a certain activity was executed. The performance and resilience uncertainties are centered on the consumption of resources like computing power or time and the proper error handling.

The causes of event-related uncertainty are similar to activity-related uncertainty. Software or human resources in a process emit events. Thus, they are the primary source of uncertainty. Examples of events are start, end, and intermediate events. The MCBPM meta-model does not further divide events. In languages like BPMN, many other semantically different event types exist for communication, compensation, or time-based logic. Referencing the meta-model and its generic events, the integrity trust concern may be caused by human resources or software that emit an event. Semantically, this uncertainty expresses whether the right event is emitted at the correct time in a process. Some events like internal escalation events need to be concealed within an organization. The carrier in the running example might not want competitors to see their internal quality assurance workflows and mitigations when something went wrong. Thus, confidentiality is another event-related trust concern that has its root on an organizational level. The availability trust concern expresses uncertainty regarding the availability of software or human resources that are needed to emit and communicate the event at the right time. In the running example, if a parcel catches fire during its delivery and nobody notices it because the supervisor is not available, it might cause severe damage. Non-repudiation of events is another fundamental uncertainty in inter-organizational workflows. For instance, one organization triggers an escalation event that starts an error-handling workflow that also includes activities to be executed by another organization. In case the organization later decides to deny that the event occurred, this might lead to

	Trust Concern	Process Element	Uncertainty Root	Question
activity-related	integrity	activity	software or resource or data	Is the activity executed correctly?
	confidentiality	activity	software or resource or data	Is the internal execution of the activity only visible to authorized resources?
	availability	activity	software or resource	Are the resources, needed for the execution of the activity available?
	non-repudiation	activity	organization	If a certain activity is performed, is it non-repudiable?
	performance	activity	software or resource	Is the activity executed within the needed border of time and resource consumption?
	resilience	activity	software or resource	Can the activity handle the case of failure of one of the involved components?
event-related	integrity	event	software or resource	Are the right events emitted from an activity?
	confidentiality	event	software or resource or data	Are the emitted events only visible to those who are authorized to see them?
	availability	event	software or resource	Are the ways to emit an event available once the event occurs?
	non-repudiation	event	organization	If an event was emitted, can the emitter deny it?
	performance	event	software or resource	Is the evaluation and emission of events executed within the right time and resource consumption constraints?

Table 3.3: Classes of uncertainties regarding trust concerns, their roots and questions to ask regarding it.

inconsistencies and re-execution of certain parts in the collaborative process. The performance uncertainty is concerned with the time it takes to trigger an event after it occurred.

Gateway-related uncertainty is concerned with whether decisions are made correctly (for exclusive splits and joins) or whether parallel behavior is coordinated correctly (for inclusive splits and joins). For competitive reasons, it might be necessary that the reasoning behind these decisions made by human resources or software is treated confidentially. Suppose anybody in the world can see why a logistics company routed a parcel through a certain hub. This would enable competitors to reverse-engineer crucial parts of the process. The reasoning for trust concerns regarding availability, non-repudiation, and performance is similar to activity-related trust concerns. Certain human or software resources are required to make decisions in a process. In case they are not available at the time in the process, this might cause a delay. Reversing decisions (i.e., violating non-repudiation) might lead to inconsistencies in coordinating different subprocesses executed by separate organizations. Performance problems (e.g., a decision takes too long) might pose a threat to the process' desired time constraints.

Process flow-related elements share a common set of relevant uncertainties. Yet, there are different semantics for intra-organizational sequence flows and inter-organizational message flows. The root causes of process-flow-related uncertainties are attributed to an organization since different actors (software or human resources) have to coordinate on an organizational level. The integrity trust concern is of relevance and deals with the right order of process elements. For example, the carrier cannot deliver a parcel before it received it. Confidentiality is regarded with respect to the privacy of the process coordination within or between companies towards non-involved outside entities. For proper execution of the process as a whole, resources that orchestrate the process-flow need to be available. The performance trust concern describes whether this orchestration is made in an acceptable time frame.

The remainder of this chapter utilizes the reference UPL for better illustration. In general, it is possible to utilize any UPL that has been created, as discussed previously, as a configuration parameter for the four main steps of Trust Mining.

3.5.5 Step 1: Uncertainty Identification

The automated identification of uncertainty is the first step of Trust Mining. For this step, Trust Mining utilizes the input process model in conjunction with the UPL and automatically annotates the process model with the uncertainty possibilities from the list. Formally, the annotation step is a function that translates an input process model p into an annotated process model p_a. This 5-tuple consists of the four process model elements from the input model and an additional uncertainty annotation mapping α. α associates every process element with a set of uncertainty possibilities.

	Trust Concern	Process Element	Uncertainty Root	Question
gateway-related	integrity	gateway	resource	Are the decisions made as desired?
	confidentiality	gateway	software or resource or data	Is the logic how decisions have been made only visible to authorized actors?
	availability	gateway	software or resource or data	Are the tools needed to evaluate a gateway available?
	non-repudiation	gateway	organization	If a certain gateway was evaluated, is it not deniable?
	performance	gateway	software or resource	Is the evaluation of gateways executed within the right time and resource consumption constraints?
process flow-related	integrity	sequence flow	organization	Are activities executed in the right order (as specified)? Is the process only terminating if it is supposed to be terminating?
	integrity	message flow	organization	Is the message flow between collaborators executed as intended? Is the process only terminating if it is supposed to be terminating?
	confidentiality	sequence flow	organization	Is the (internal) sequence flow and all associated data objects only visible to authorized actors?
	confidentiality	message flow	organization	Is the message flow and all associated data objects only visible to authorized actors?
	availability	sequence flow	organization	Is everything needed to coordinate activities intra-organizationally available?
	availability	message flow	organization	Is everything needed to coordinate activities inter-organizationally available?
	performance	sequence flow	organization	Is the flow between activities (intra-organizationally) conducted within the right time and resource consumption constraints?
	performance	message flow	organization	Is the flow between activities (inter-organizationally) conducted within the right time and resource consumption constraints?

Table 3.4: Classes of uncertainties regarding trust concerns, their roots and questions to ask regarding it (continued).

Algorithm 2: annotateUncertainties

Input: $p = (N, E, L, \lambda), UPL$
Result: p_a
$\alpha(x) := \{\}, \forall x \in N \cup E$;
foreach $n \in N$ **do**
  foreach $up = (r, tc, et) \in UPL$ **do**
    if $\tau(n) == et$ **then**
      $\alpha(n) := \alpha(n) \cup up$;
    end
  end
end
foreach $e \in E$ **do**
  foreach $up = (r, tc, et) \in UPL$ **do**
    if $\eta(e) == et$ **then**
      $\alpha(n) := \alpha(n) \cup up$;
    end
  end
end
$p_a := (N, E, L, \lambda, \alpha)$;
return p_a

$$\begin{aligned} p_a &= (N, E, L, \lambda, \alpha) \\ \alpha &: N \cup E \rightarrow UP \subseteq UPL \end{aligned} \tag{3.15}$$

The annotation is illustrated in Algorithm 2. To build the annotation mapping, the algorithm iterates over every node $n \in N$ and every edge $e \in E$ of the input process model p. For each of these elements, it iterates over all uncertainties $up = (r, tc, et) \in UPL$. If the type of the process element ($\tau(n)$ or $\eta(e)$ respectively) matches the element type et of the current uncertainty, then the algorithm associates this uncertainty possibility with the process element.

Figure 3.11 illustrate the uncertainty annotated process model for a portion of the delivery example. The "create report" activity is a user activity (marked with the small user icon in BPMN). Hence, for this activity, a human resource (Res.) executes a task assisted by software (Soft.). These two entities cause uncertainty regarding integrity (Int.), confidentiality (Conf.), availability (Av.), performance (Perf.), and resilience (Resl.). The organization causes uncertainty regarding the non-repudiation, as defined in the reference UPL.

3.5.6 Step 2: Dependency Analysis

In its first step, Trust Mining annotated the input model with uncertainty possibilities. This annotation identifies uncertainties that are purely encapsulated within one process element. Yet, other uncertainties may exist that result from process-related relationships between different process collaborators that cannot be attributed to a single process element. Thus, Step 2 analyzes the process model for cross-organizational dependencies in the process. Step 2 distinguishes process dependencies that are *data dependencies* or *message dependencies*.

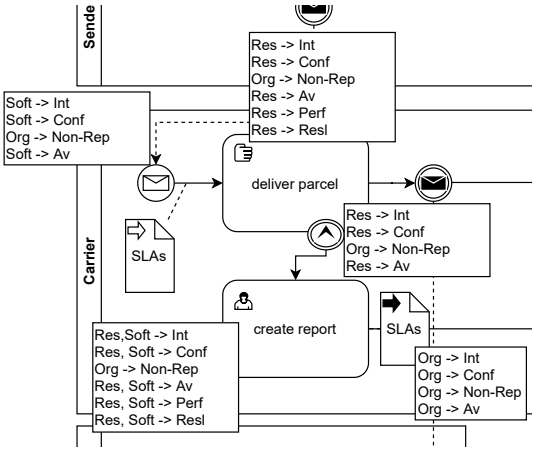

Figure 3.11: Illustration of the annotated model. This diagram shows a small portion of the BPMN diagram of the parcel delivery case in Figure 3.9 annotated with the reference uncertainty possibilities defined in in Tables 3.3 and 3.4. The full annotated graph includes many more annotations; this illustration only depicts a subset of the graph for simplicity.

The outcomes of this step are two dependency graphs. These graphs and the uncertainty-annotated model are the input to Step 3 of Trust Mining.

Data Dependency When different parties access shared data in a process, the integrity of this data is essential for the correct execution of activities. A collaborator executing a task based on data delivered by another collaborator poses an uncertainty. However, in contrast to the local uncertainties identified in Step 1, this uncertainty is not entirely caused by the executing, i.e., data consuming, organization. The origin of the data, i.e., the data delivering organization, also causes uncertainty *transitively*. Therefore, data dependency analysis is a method to analyze such relationships in a process.

Data objects model inputs and outputs for elements like activities in the process. BPMN adopts data objects through input data objects, output data objects, and plain data objects. Other process modeling languages, like Petri nets, for instance, can support data flow modeling and data operations with special markings [162]. For Trust Mining, it is required that all data objects are either input or output objects, reflecting the definition in the MCBPM meta-language. The semantics of plain data objects is not expressive enough to analyze data dependency relationships. Additionally, it is also required that the same data objects have the same name so that Trust Mining can automatically execute the analysis.

Data dependencies become relevant in situations in a process where one organization is consuming data that another organization is producing, modifying, or forwarding. In the running example of supply chain management, the receiver obtains the SLA data object from the carrier. This document states which SLAs the carrier needs to adhere to during the process. It originates from the sender who has initially transferred it to the carrier. The receiver is in a situation with two dependencies regarding the SLA data object. First, the receiver depends on the sender to specify the correct SLAs. If the receiver bought fireworks in

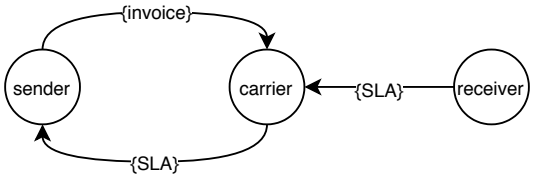

Figure 3.12: Data dependency graph of the running example.

the sender's e-commerce store and the sender misses to specify that the parcel needs to be in a non-static environment, the parcel might explode. Second, the receiver also relies on the carrier not to tamper with the data object before forwarding and adhering to it. This situation constitutes a data dependency.

Trust Mining conceptualizes this dependency in a relationship graph. Formally, the data dependency graph p_{dep}^{data} consists of nodes and edges. In this graph, the swimlanes $l \in L$ of the process model p are the nodes. Edges connect the swimlanes in the graph in a way that p_{dep}^{data} has an edge (l_i, l_j) for every pair of lanes, where at least one path from a data input to a data output for different organizations exists regarding the same data object. This means that every connection in that graph symbolizes at least one situation where an organization consumes data as an input that another organization produces as an output. The mapping $do()$ associates the data object to the edge.

$$\begin{aligned} p_{dep}^{data} =& (L, E_{dep}^{data}), \forall e_{dep}^{data} = (l_i, l_j) \in E_{dep}^{data} : \\ & \exists((n_i, n_k), ..., (n_l, n_j)), n_i, n_k, n_l, n_j \in N : \\ & \tau(n_i) = \text{data-in} \wedge \tau(n_j) = \text{data-out} \\ & do(n_i) = do(n_j) \wedge \lambda(n_i) \neq \lambda(n_j) \end{aligned} \tag{3.16}$$

Additionally, the edges of the data dependency graph may be enhanced with definitions of the data objects that cause the relationship. Therefore, Trust Mining introduces *delta data sets* δ_{l_j,l_i}^{data}. These delta sets describes the set of data objects that a lane l_j depends on as inputs that l_i produces as output.

$$\delta_{l_j,l_i}^{data} = \{dataObject_1, ..., dataObject_n\} \tag{3.17}$$

The data dependency graph may be annotated with the δ^{data} sets as seen in Figure 3.12. It illustrates the dependency graph from the delivery example. The carrier consumes the SLA data object as an input that the sender produced as an output. This creates an edge in the data dependency graph from the sender to the carrier. The carrier forwards the SLA object to the receiver and thus creates a data dependency from the receiver to the carrier regarding the SLA object. After the conclusion of the delivery, the carrier sends an invoice data object to the sender. This creates a data dependency from the sender to the carrier regarding the invoice. In general, one edge can have multiple data objects in its dependency set. Formally, the data dependency graph in the example can be described as follows:

Algorithm 3: createDataDependencyGraph

Input: $p_a = (N, E, L, \lambda, \alpha)$
Result: p_{dep}^{data}, δ^{data}

$E_{dep}^{data} := \{\}$;
$\delta^{data} := \{\}$;
foreach $l_j \in L$ **do**
  foreach $l_i \in L$ **do**
    $\delta_{l_j,l_i}^{data} := \{\}$;
  end
end
foreach $n_j \in N$ **do**
  if $\tau(n_j)$ == *data-in* **then**
    foreach $n_i \in N$ **do**
      if $\tau(n_i) ==$ *data-out*, $do(n_i) == do(n_j), \lambda(n_i) \neq \lambda(n_j), n_j \in$ *depthFirstSearch*(n_i) **then**
        $e := (\lambda(n_j), \lambda(n_i))$;
        $do(e) := do(n_i)$;
        $\delta_{\lambda(n_j),\lambda(n_i)}^{data} := \delta_{\lambda(n_j),\lambda(n_i)}^{data} \cup do(e)$;
        $E_{dep}^{data} := E_{dep}^{data} \cup e$;
      end
    end
  end
end
foreach $l_j \in L$ **do**
  foreach $l_i \in L$ **do**
    $\delta^{data} := \delta^{data} \cup \delta_{l_j,l_i}^{data}$;
  end
end
$p_{dep}^{data} = (L, E_{dep}^{data})$;
return p_{dep}^{data}, δ^{data}

$$
\begin{aligned}
p_{dep}^{data} =& (\{\text{sender, carrier, receiver}\}, \{(\text{sender, carrier}),\\
&(\text{receiver, carrier}), (\text{carrier, sender})\})\\
\delta_{sender,carrier}^{data} =& \{\text{invoice}\}\\
\delta_{carrier,sender}^{data} =& \{\text{SLA}\}\\
\delta_{receiver,carrier}^{data} =& \{\text{SLA}\}
\end{aligned}
\tag{3.18}
$$

Procedurally, the data dependency graph may be created according to Algorithm 3. The algorithm consumes the annotated process model p_a as input and produces p_{dep}^{data} and the data annotation δ^{data} as an output. This algorithm's main objective is to identify the edges that need to be included in the data dependency graph. Therefore, the algorithm iterates over all $n_i \in N$ that are data input objects. For each of these data input objects, the algorithm traverses all other nodes in the process graph. If the element is a data output element that is in a different swimlane than the current input element but has the same data object, the algorithm performs a depth-first search. Depth-first search leaves

an ordered sequence of nodes that are reachable from the data input. If the data output is in this sequence, there is a path from the input to the output object. In this case, the algorithm creates a new edge between the two lanes and adds the edge to the set edges of p_{dep}^{data}. In the end, the algorithm collects all data dependency object descriptions in the δ^{data} and returns it together with the graph p_{dep}^{data}.

Message Dependency Additionally to data dependency, message exchanges between different collaborators also pose another type of uncertainty. Similar to data dependency, they can also not be attributed to a single process component in isolation. Message dependencies result from the uncertainty in the exchange of a message that an organization sends and another one receives. Thus, it is necessary to analyze relationships to identify message-related uncertainties. In contrast to data dependency, some messages do not have data objects associated with them. Messages may not be triggered as intended, contents associated with the message may be corrupted, or the workflow may not continue as desired.

In the MCBPM meta-model, message flows are modeled through edges. In the running example in Figure 3.9, the data message flow between carrier and receiver deals with the handover of a physical object. The message exchange can be seen as an interaction between the two organizations through their associated human resources. Uncertainty regarding the handover message exchange may be concerned with whether the right parcel is transferred. Furthermore, the question of whether all information included in the message is correct poses a message-related uncertainty.

In formal notation, the message dependency graph p_{dep}^{msg} consists of the lanes $l \in L$ as nodes and edges E_{dep}^{msg} for every edge in the process model p_α between two different lanes. Edges between different swimlanes always express message flows between separate organizations.

$$\begin{aligned} p_{dep}^{msg} =& (L, E_{dep}^{msg}) \\ e_{dep}^{msg} =& (l_i, l_j) \in E_{dep}^{msg} : \\ & \exists e = (n_i, n_j) \in E : \\ & \lambda(n_i) \neq \lambda(n_j) \end{aligned} \tag{3.19}$$

Visually, the message dependency graph looks similar to the data dependency graph. The most significant difference is that the edges' labels do not denote the data objects as δ^{data} does. Instead, the labels δ^{msg} denote the carnality of the message flows between the two lanes.

$$\delta_{l_i,l_j}^{msg} = |e = (n_i, n_j) \in E : \lambda(n_i) = l_i, \lambda(n_j) = l_j| \tag{3.20}$$

The running example has message flows between sender and carrier, carrier and receiver, receiver and carrier, and carrier and sender. Between these pairs,

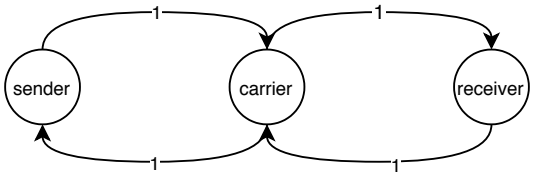

Figure 3.13: Message dependency graph of the running example.

Algorithm 4: createMessageDependencyGraph

Input: $p_a = (N, E, L, \lambda, \alpha)$
Result: p_{dep}^{msg}, δ^{msg}

$E_{dep}^{msg} := \{\}$;
$\delta^{msg} := \{\}$;
foreach $l_j \in L$ **do**
  foreach $l_i \in L$ **do**
    $\delta_{l_j,l_i}^{msg} := 0$;
  end
end
foreach $e = (n_j, n_i) \in E$ **do**
  if $\lambda(n_j) \neq \lambda(n_i)$ **then**
    $e := (\lambda(n_j), \lambda(n_i))$;
    $\delta_{\lambda(n_j),\lambda(n_i)}^{msg} := \delta_{\lambda(n_j),\lambda(n_i)}^{data} + 1$;
    $E_{dep}^{msg} := E_{dep}^{msg} \cup e$;
  end
end
foreach $l_j \in L$ **do**
  foreach $l_i \in L$ **do**
    $\delta^{msg} := \delta^{msg} \cup \delta_{l_j,l_i}^{msg}$;
  end
end
$p_{dep}^{msg} = (L, E_{dep}^{msg})$;
return $p_{dep}^{msg}, \delta^{msg}$

there is exactly one message exchange each. Thus, all cardinalities are 1. The graph can be seen in Figure 3.13.

The message dependency graph of the running example is structured as follows:

$$\begin{aligned} p_{dep}^{msg} = (&\{\{(\text{sender,receiver,receiver})\}\},\\ &\{(\text{sender, carrier}), (\text{carrier, receiver}),\\ &(\text{receiver, carrier}), (\text{carrier, sender})\}) \end{aligned} \tag{3.21}$$

The message cardinalities of the example are structured as follows:

$$\begin{aligned} \delta_{\text{carrier, sender}}^{msg} =& 1, \delta_{\text{receiver, carrier}}^{msg} = 1,\\ \delta_{\text{carrier, receiver}}^{msg} =& 1, \delta_{\text{sender, carrier}}^{msg} = 1 \end{aligned} \tag{3.22}$$

Procedurally, the construction of the message dependency graph is shown in Algorithm 4. The algorithm utilizes the process model as an input and produces the message dependency graph p_{dep}^{msg} and the cardinality annotation set δ^{msg} as an output. Initially, there are no edges. Thus all pairs have a connection of 0. Afterward, the algorithm iterates over all edges of the process model. When it detects an edge between two different lanes, i.e., if $\lambda(n_j) \neq \lambda(n_i)$, then an edge is added if it is not existing and the cardinality is increased. After the algorithm traversed the full process model, the message dependency graph and combined cardinality annotation set δ^{msg} are returned.

3.5.7 Step 3: Uncertainty Metrics

Trust Mining's third step uses the uncertainty-annotated business process model from Step 1 and the process dependency graphs from Step 2 as inputs. This step applies aggregation of uncertainties and trust relationships to generate trust-related metrics. In general, the defined metrics revolve around the following questions:

- How much uncertainty is present in the process?
- Which collaborator is responsible for which uncertainty?
- How dependent are collaborators on uncertain process elements executed by other collaborators?

The following sections introduce different metrics. All metrics utilize characteristics of the uncertainty-annotated business process model and the process dependency graphs. Furthermore, they can be clearly related to the TRUB trust model and answer the questions above.

Global Process Uncertainty On a general level, process engineers and analysts want to understand how much uncertainty is present in a process as a whole. A metric that enables this analysis on a global level may be utilized to compare different processes to each other. Therefore, the *global process uncertainty (GU)* is a quantification of uncertainty in the whole process. Conceptually, GU is defined as the sum of all possible uncertainties across all elements in all swimlanes.

Formally, GU of an annotated process model p_a is defined as the sum of the different α-mapping set sizes.

$$GU(p_a) = \sum_{i \in N \cup E} |\alpha(i)| \tag{3.23}$$

In the running example, GU is 172.

The GU metric illustrates the absolute uncertainty that is present in the whole process. This leads to some characteristics that pose a threat to GU's utility for trust analysis. Generally, the more elements a process has, the more uncertainty possibilities may exist. In detail, the GU metric varies depending on the concrete elements used. Thus, processes with largely different numbers of process

elements cannot be compared. Therefore, Trust Mining introduces the *average element uncertainty (AEU)* metric as a *relative* uncertainty measure per process element. The AEU is obtained by dividing GU by the number of process elements in the whole process.

$$AEU(p_a) = \frac{GU(p_a)}{|N \cup E|} \tag{3.24}$$

In the supply chain example, the AEU is $3.\overline{90}$.

Lane Uncertainty The *lane uncertainty (LU)* metric shows how much uncertainty is present within the subprocesses of certain process collaborators. Trust Mining distinguishes between *absolute lane uncertainty (ALU)* and *relative lane uncertainty (RLU)*. For the formal definition of ALU and RLU, this section introduces a helper function $isLane(i, l)$. This function returns 1 if a certain process element i belongs to a lane l and 0 otherwise.

$$isLane(i, l) = \begin{cases} 1, & \text{if } \lambda(i) = l \\ 0, & \text{otherwise} \end{cases} \quad i \in N \cup E, l \in L \tag{3.25}$$

Thus, Trust Mining defines the ALU as the sum of all uncertainty possibilities that a certain organization (modeled through a swimlane) is responsible for.

$$ALU(p_a, l) = \sum_{i \in N \cup E} isLane(i, l) \cdot |\alpha(i)| \tag{3.26}$$

The RLU is the relative version of ALU. The RLU is a real value between 0 and 1 that is computed dividing the ALU of a specific swimlane with the GU. Thus, this metric can visualize that a particular organization is responsible for a certain percentage of all uncertainties in a process.

$$RLU(p_a, l) = \frac{ALU(p_a, l)}{GU(p_a)} \tag{3.27}$$

In the delivery example in Figure 3.11, the sender collaborator has an RLU of 0.25, the carrier has 0.407, and the receiver has 0.343. The metric indicates that the carrier has (relatively) more uncertainties in its domain of influence than the receiver.

Uncertainty Balance The lane uncertainty metrics provide isolated insights on the proportion of uncertainty that resides within different collaborators' influence domains. The interpretation of these metrics depends on the number of process collaborators. When a certain organization has an RLU of 0.2 in a process with five collaborators, this organization has an average significance for the process. However, if it has an RLU of 0.2 in a process with two collaborators,

this means that the other organization is responsible for 80% of all uncertainties in the process. This characteristic makes the interpretation of the RLU in the context of a process difficult.

Thus, Trust Mining defines *uncertainty balance* metrics to enable the comparison between different process collaborators. Assuming that the global uncertainty in a process was equally distributed among all $|L|$ collaborators, each of them would be responsible for $1/|L|$ uncertainties. Every deviation from this indicates an *imbalance*. Therefore, Trust Mining introduces the *lane uncertainty balance (LUB)* to identify deviations from such a perfectly balanced scenario. Formally, LUB is defined as follows:

$$LUB(p_a, l) = RLU(p_a, l) - \frac{1}{|L|} \tag{3.28}$$

In a completely balanced process with three organization collaboration, every swimlane would be responsible for $1/3$ of the uncertainties. Assuming perfect balance, LUB would be 0 for all collaborators. Deviations from 0 in the LUB metrics indicate uncertainty imbalances. Assuming a lane l is responsible for $2/3$ of all uncertainties, then $LUB(p_a, l) = +1/3$. In a perfectly balanced scenario, l would be responsible for $1/3$ of all uncertainties. The deviation from this perfect balance is $2/3 - 1/3 = +1/3$ for the organization. On the other hand, if a collaborator is responsible for fewer uncertainties, then the uncertainty balance is negative, i.e., $LUB(p_a, l) \leq 0$. In case that a lane l' is only responsible for $1/6$ of the uncertainties in a process with three collaborators, then $LUB(p_a, l) = 1/6 - 1/3 = -1/6$.

Cross-Lane Uncertainty Dependencies The previously discussed metrics have their focus on *atomic* uncertainties without considering the process relationships between actors. Thus, the following paragraphs introduce *cross-lane uncertainty dependency* metrics. These metrics utilize the data dependency graph p_{dep}^{data} and message dependency graph p_{dep}^{msg} as a base. The cross-lane uncertainty dependency metrics utilize the in-degree and out-degree of nodes.

In p_{dep}^{data}, an edge (l_i, l_j) expresses that the organization behind lane l_i uses a data object as an input, which originates from lane l_j. Hence, l_j may tamper with the integrity of the data object, send the wrong data object, or execute any other undesired behavior. This constitutes a data dependency. Formally, a situation where many other lanes consume data from one single lane l_j can be described with a large in-degree deg^{in} of this specific lane l in p_{dep}^{data}. This circumstance indicates that l has a large *data influence (DI)* on other lanes.

$$DI(p_{dep}^{data}, l) = deg^{in}_{p_{dep}^{data}}(l) \tag{3.29}$$

Equivalently, a large out-degree deg^{out} of a lane l expresses a lane's strong dependency on data from other collaborators. Trust Mining introduces this metric as the *data dependency (DD)* of a lane.

$$DD(p_{dep}^{data}, l) = deg^{out}_{p_{dep}^{data}}(l) \tag{3.30}$$

Regarding message dependency, Trust Mining introduces similar metrics. If a lane l has a large in-degree deg^{in} in the message dependency graphs p_{dep}^{msg} this means that this lane has a large *message influence (MI)* on other lanes.

$$MI(p_{dep}^{msg}, l) = deg^{in}_{p_{dep}^{msg}}(l) \tag{3.31}$$

Equivalently, when l has a large out-degree deg^{out}, this indicates dependency on messages and associated objects from other collaborators. Thus, this metric is defined as the *message dependency (MD)* of a lane.

$$MD(p_{dep}^{msg}, l) = deg^{out}_{p_{dep}^{msg}}(l) \tag{3.32}$$

In the running example, the carrier has incoming and outgoing message flows with each of the other two collaborators. This results in a message dependency and message influence of two for both metrics ($MD = 2, MI = 2$).

3.5.8 Step 4: Relevancy Analysis

Trust Mining's first three steps analyzed trust properties of a business process globally. Step 4 compares these properties to the *trust tolerance profiles* of different *trust personas*. Trust personas can be any process stakeholders or collaborators with interest in the trust-aware execution of the process. They have different trust relationships and preferences with collaborators. The comparison in Step 4 aims to identify which trust issues are relevant from the *perspective* of a specific trust persona. Therefore, trust personas express their tolerance profiles with *trust policies*. These policies are input parameters for the relevancy analysis.

Trust Personas Generally, relevancy analysis aims to assess *who trusts whom for what*. According to the schema introduced by Khare et al., a *trustor* trusts a *trustee* for a certain *trust subject* [148]. In Trust Mining, any entity interested in the trust properties and the trustworthiness of a business process can be a trust persona. Thus, a trust persona is a trustor in a trust relationship.

In the running example of the transport of sensitive goods as depicted in Figure 3.9, all organizations collaborating in the process can be trust personas. They have different interests in the trust properties of the process.

- *Sender and Receiver*: The sender and receiver are interested in the trust properties of the process because they want the parcel to be delivered as desired. If the parcel explodes on the way or never reaches its destination, this might cause severe economic damage to both of them.

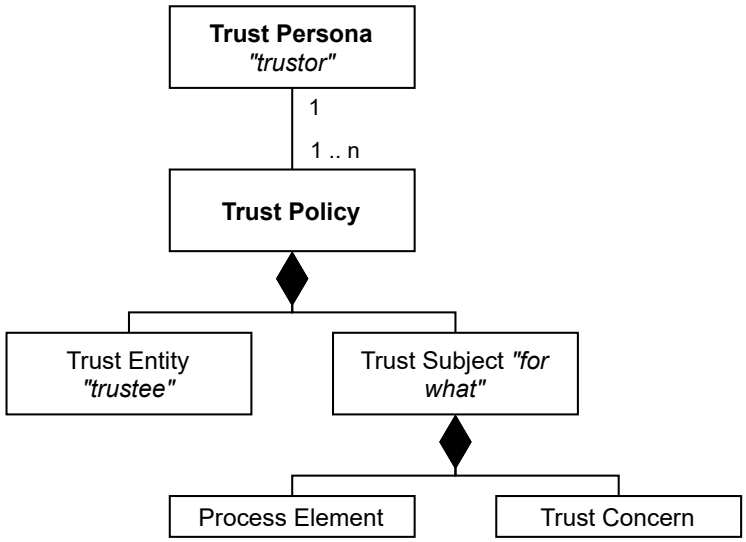

Figure 3.14: Model for trust policies. One trust persona (trustor) has n trust policies. Policies describe a trust entity and a trust subject.

- *Carrier*: The carrier is interested in the trust properties of the process because the carrier wants to receive reimbursement for the provided services. Furthermore, the carrier is interested in having long-term customers that bring recurring revenue. Therefore, it is also important to establish the trustworthiness of the customers regarding payments.
- *External Trust Personas*: In addition to entities that are directly involved in the process as collaborators, other process stakeholders can be trust personas. For instance, it might be necessary for government organizations to verify that regulatory and security guidelines are not violated and the delivery does not pose a public safety issue.

Trust Policies Trust policies define a trust persona's trust tolerance profile. These policies are inspired by elements from the trust model proposed by Grandison and Sloman [144] and the same-named concept defined by Khare et al. [148].

One trust persona has n trust policies, as illustrated in Figure 3.14. The trust policies define for a trust persona (trustor), which entities (trustees) the persona trusts for a certain trust subject. Trust entities are all organizations that are included in the process model with a swimlane. The trust subject consists of at least one process element and at least one trust concern. In the running example, the sender (trust persona) trusts the receiver (trust entity) that the evaluation of the condition of a parcel during the handover (gateway, process element) is always done correctly (integrity, trust concern). However, the sender might not trust the carrier that the parcel does not explode.

Trust policies do not distinguish between different "degrees" of trust, meaning different extents of trust in a single trust subject. The general concept defines that a trust persona either trusts a trust entity for a certain trust subject or not. Thus, on this fine-granular level, trust policies describe not a probability but a boolean value. This concept of fine-granular utilizing trust policies is

an alternative to the principles in risk-aware business process management. Risk-aware BPM always assigns a risk probability to a risk. In many situations, it is not possible to determine this probability. Especially in inter-organizational processes where one organization does not have access to data needed to determine the risk probability, the concept of trust policies in Trust Mining provides a viable alternative. In Trust Mining "partial" trust does not exist on the atomic level. However, trust policies (or the absence thereof) can express partial trust from a process-wide perspective. A trust persona may trust a collaborator for one activity but not for another one.

Trust Mining defines trust policies as a *positive* trust relationship. A trust policy expresses that a trust persona trusts a trust entity fully with respect to a trust subject. In case the trust persona does not trust a particular trust entity for a trust subject, there is no trust policy associated with it. Subsequently, this leads to the semantics that a trust persona does not trust any combination of trust entity and trust subject that is not subject to a (positive) trust policy. All trust policies for a trust persona represent her *trust tolerance profile*.

Table 3.5 shows an example of a trust policy definition for a trust persona. The entries of the table are classified according to the respective fields of the trust policies:

- Trust Persona: This overall classification illustrates to which trust persona the trust policies are associated (trustor). Thus, it can be any process collaborator or stakeholder.
- Trust Entity: The trust entity is the trustee in a trust policy. Thus, this field can be *all* organizations as represented in the lanes in the model or any subset of process collaborators.
- Process Element: Process elements and trust concerns describe the trust subject, i.e., for what a trust persona trusts a trust entity. Thus, this field can be *all* elements, classes of process elements, e.g., manual tasks, or specific instances of process elements, e.g., the create invoice script task.
- Trust Concern: The trust concern is the second part of the composite trust subject. In the table, it can be *all* trust concerns or a specific subset of the previously defined trust concerns.

Formally, Trust Mining defines a trust policy pol as a triplet of one or more trust entities, process elements, and trust concerns. A set of trust policies associated with one trust persona $pers$ represents her trust tolerance profile TTP.

$$\begin{aligned} TTP_p^{pers} &= \{pol_1, ..., pol_n\} \\ pol_i &= (L', ET', TC') \\ L' &\subseteq L \\ ET' &\subseteq ET \cup N \cup E \\ TC' &\subseteq TC \end{aligned} \tag{3.33}$$

Trust Persona	sender	
Trust Entity	**Trust Subject**	
	Process Element	**Trust Concern**
sender	all	all
receiver	all	all
all	sequence flow	all
all	message flow	all
carrier	create invoice	confidentiality
carrier	handover parcel	all
carrier	send invoice	all

Table 3.5: Trust policies of the sender in the running example of the delivery of dangerous goods.

Algorithm 5: reduceUncertaintiesForPerspective(ϱ)

Input: $p_a = (N, E, L, \lambda, \alpha), TTP_p^{pers}$
Result: p_r^{pers}
$p_r^{pers} := p_a$;
foreach $ne \in N \cup E$ **do**
  foreach $pol = (L', ET', TC') \in TTP_p^{pers}$ **do**
    if $\lambda(ne) \subseteq L', ne \subseteq ET'$ **then**
      foreach $UP = (r, tc, et) \in \alpha(ne)$ **do**
        if $tc \subseteq TC'$ **then**
          $\alpha(ne) := \alpha(ne) \setminus UP$;
        end
      end
    end
  end
end
return p_r^{pers}

Trust Assessment Trust Assessment is the last part of Step 4. Its goal is to analyze which trust concerns in the process model are relevant for specific trust personas. Therefore, this task *reduces* the annotated business process model with the possible uncertainties based on the trust policies.

Formally, p_r^{pers} describes a process model with annotated reduced trust issues as an annotated model. The ϱ function describes the reduction of an annotated process model p_a with the trust policies of a trust persona TTP_p^{pers}.

$$p_r^{pers} = \varrho(TTP_p^{pers}, p_a) = (N, E, L, \lambda, \alpha') \tag{3.34}$$

Algorithm 5 explains the functionality of the ϱ-function. Initially, the algorithm initializes p_r^{pers} with p_a. Afterward, the algorithm iteratively traverses all process elements ne and analyzes the coverage of an annotated uncertainty with a trust policy. In case there exists a trust policy pol in the trust tolerance profile of that persona TTP_p^{pers} that matches the type of the process element, the lane, and the annotated trust concern, the algorithm removes it from p_r^{pers}.

In the running example, the sender trusts the receiver fully for every part of

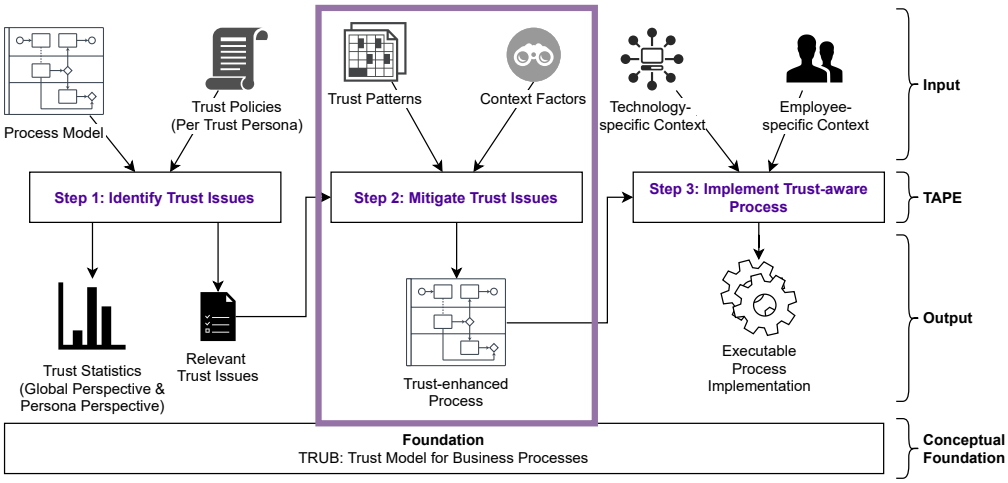

Figure 3.15: Mitigating trust issues in the context of TAPE.

the process that the receiver is involved in. Hence, the reduced model does not include any uncertainties in that lane from the perspective of the sender trust persona.

To analyze the process model from the perspective of a trust persona, the metrics from Step 3 are applied to p_r^{pers}. Each of the trust personas has a separate trust tolerance profile and trust a different p_r^{pers}. The metrics applied to the reduced uncertainty-annotated model provide a specific trust persona a detailed overview of the *relevant* trust issues from the *persona's point of view*.

3.6 Mitigating Trust Issues in Business Processes

Mitigating trust issues in a process is the second step in the TAPE method. This step utilizes the list of relevant trust issues from Step 1 and systematically identifies counteractions. Therefore, three different strategies may be used[5].

Reducing uncertainty in a process aims to lower the probability *that* something undesired happens. Thus, reducing uncertainty is a *proactive* approach. Consider the example of delivering dangerous goods, which brings uncertainty whether the carrier handles the parcel carefully enough. The sender of fireworks might be worried that their fireworks would explode during the delivery. Every measure that decreases the probability that an undesired action happens can be considered reducing uncertainty. For example, allocating a dedicated person in the delivery process who purely focuses on that specific parcel reduces the uncertainty that something undesired happens.

In many cases, it is not possible to eliminate uncertainty completely. Therefore, the second strategy becomes relevant. *Reducing vulnerability* of a process aims to reduce the impact for the process *when* something undesired happens. Thus, reducing vulnerability is a *reactive* approach. In the example of the delivery

[5]Mitiging trust issues with trust patterns has been published in the following conference paper: M. Müller M., N. Ostern, M. Rosemann (2020) Silver Bullet for All Trust Issues? Blockchain-Based Trust Patterns for Collaborative Business Processes. In: Asatiani A. et al. (eds) Business Process Management: Blockchain and Robotic Process Automation Forum. BPM 2020. Lecture Notes in Business Information Processing, vol 393. Springer, Cham.

of dangerous goods, a compensation mechanism could alleviate the monetary loss of the receiver in case the firework parcel explodes.

Building confidence is the third mitigation strategy. The strategy aims to increase the *perceived* trustworthiness of a process. This is achieved by adding external *sources of trust* to a process. External sources are not directly a part of the process. An example of an external source of trust is a reputation system. Before the sender, in the delivery example, decides to select a certain carrier for the job, the sender might use an online rating service. Online rating services illustrate the experiences of other customers with a specific customer. For instance, a five-star rating might indicate that other customers had good experiences with the carrier. This increases the perceived trustworthiness of the carrier and builds confidence.

3.6.1 A Taxonomy for Trust Patterns

This section describes a taxonomy of patterns that enhance trust-related aspects of a process. A *trust pattern* is a *design pattern* that aims to increase trust-realated attributes of a process. In software design, design patterns are reusable solutions to common problems that occur within certain situations [163]. Trust patterns have the same aim. They are domain-independent solutions to trust issues in a business process.

In the context of the TAPE method, trust patterns aim to solve trust issues that have been identified in the list as an output of Step 1. The following presents a taxonomy that can classify different trust patterns in a way that they can be clearly related to these trust issues. The concept has been established following the methodology as described by Nickerson et al. [164]. The taxonomy consists of five dimensions.

Dimension 1: Trust-enhancing Strategy The three strategies for mitigating trust issues and thus enhancing trust include reducing uncertainties, reducing process vulnerabilities, or building confidence.

Dimension 2: Uncertainty Root This dimension describes the origin of the trust issues. Thus, this is the root of uncertainty of the TRUB model.

Dimension 3: Trust Concern Similar to uncertainty roots, this dimension represents the trust concerns of the TRUB model. The descriptions in the following sections utilize TC_{PR} as set of trust concerns (integrity, confidentiality, availability, non-repudiation, and performance).

Dimension 4: Process Component This dimension describes in which elements of the process the trust concern is of relevance. The following discussion utilize the elements of MCBPM as process components.

Dimension 5: Limitations Finally, this dimension describes the limits of a trust pattern.

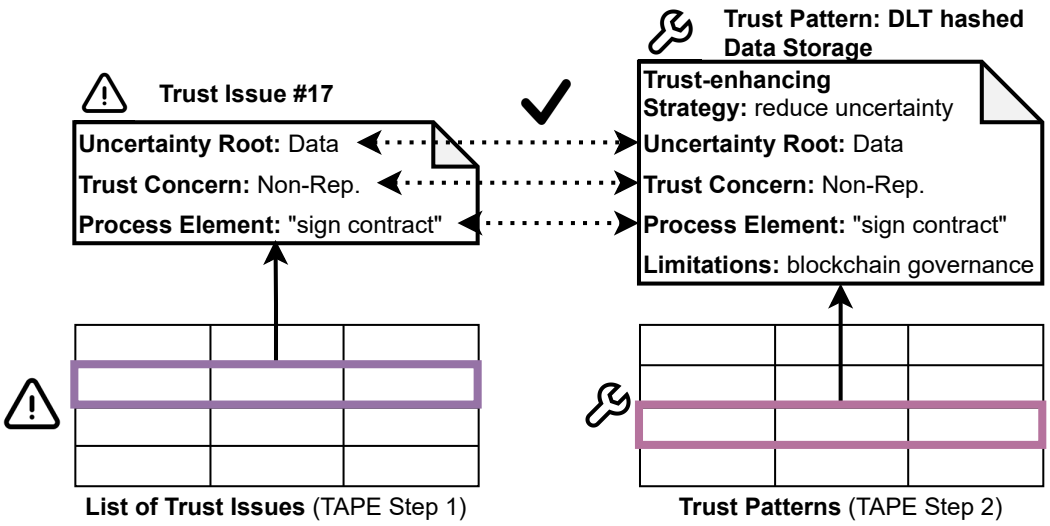

Figure 3.16: Trust patterns can be matched to trust issues as mitigations.

The taxonomy incorporates elements of the TRUB model. The list of identified trust issues in step one of the TAPE method is also based on TRUB. Thus, it is possible to match the taxonmoy's dimensions and the trust issues as illustrated in Figure 3.16.

3.6.2 The Role of Distributed Ledger Trust Patterns

In principle, there are many artifacts that can be used to enhance trust in collaborative business processes. Distributed ledgers are especially suited to mitigate trust-issues due to their inherent trust-enhancing capabilities. These capabilities derive from the fundamental setup of a DLT. Distributed ledgers let different parties share and modify data records in a tamper-proof way [47]. Data records can be created or modified with transactions. Transactions are linearly ordered. In that way, it is possible to replay all data changes from the beginning and verify the current state of particular pieces of information. All data records and transactions together create a global state at any point in time. State changes are inflicted through transactions. The parties engaging in the distributed ledger system interact with each other in a peer-to-peer network. They maintain consensus on the accepted transactions and their order. This mechanism ensures consistency of the current state across all participants [48].

Thus, distributed ledgers are mostly suited to mitigate trust issues regarding *integrity*, *non-repudiation*, and *availability*. The following collection discusses a set of trust patterns without the claim for completeness. The focus of this work is the assessment of the suitability of DLTs to mitigate trust issues. Thus, the following sections provide deeper focus on DLTs. Trust patterns that do not involve a distributed ledger are sketched at the end of each sub-section.

Running Example The next sections will explain each pattern in general and illustrate it referencing the same running example of the supply chain of dangerous goods in Figure 3.17.

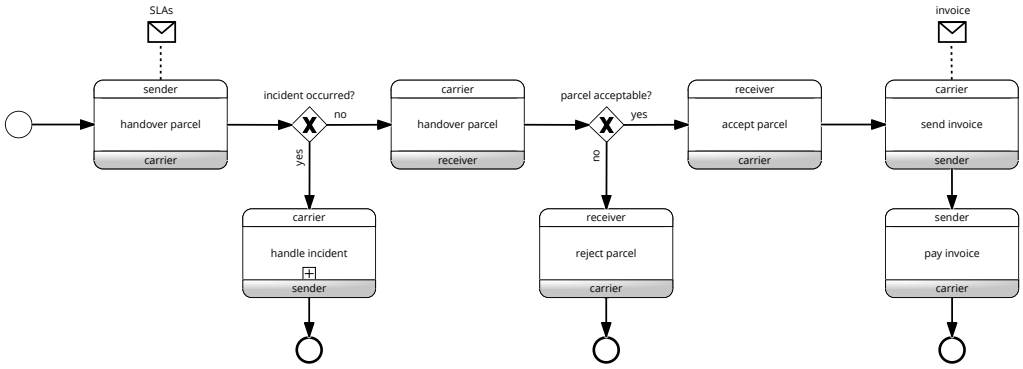

Figure 3.17: The supply chain of dangerous goods running example modeled as a BPMN choreography diagram.

Trust Pattern Modeling The next sections utilize BPMN 2.0 to describe the functionality of the trust patterns in detail. The focus is on trust patterns involving distributed ledgers. Therefore, a way to model the role of a DLT in the context of a business process is a necessity. In current BPM literature, there is no consensus on how to incorporate distributed ledgers in process models. Thus, the next paragraphs provide an overview of recent approaches to model distributed ledgers in collaborative business processes.

Ladleif et al. propose an extension of BPMN 2.0 choreography diagrams to accommodate the semantics of distributed ledgers [127]. BPMN 2.0 choreography diagram models the interactions between two or more organizations that aim to achieve a common business goal [38]. These types of diagrams focus on interfaces and communication between the process participants. The details of the work executed autonomously by the different process participants are not the primary focus of choreography diagrams. From an outside view, BPMN choreography diagrams represent business agreements (or contracts) and the expected behavior of the process collaborators. Figure 3.17 shows the initial version of the running example of a supply chain of dangerous goods modeled as a choreography diagram. The diagram emphasizes the interactions between the different organizations. This makes it easy to understand interfaces and message exchanges. However, parts of the process that a single organization executes autonomously are not shown. In a trust-centered view on a process, this is essential. In situations where a part of the process is executed as a private subprocess without any interaction are important to analyze. Yet, these situations are not represented in choreography diagrams. Thus, this type of diagram is not suited for a trust-centered view.

BPMN 2.0 collaboration diagrams are a popular way to illustrate distributed ledgers in process models. López-Pintado et al. assume that process collaborators use a common execution infrastructure, i.e., a distributed ledger with smart contracts [130]. They argue that the participants interact with the DLT as if they were virtually in the same organization. Thus, the model looks like n intra-organizational process that uses a traditional BPMS. The semantics of the inter-organizational aspect are hidden within the elements that depict interaction with the ledger. Thus, the unique role of the ledger is modeled implicitly.

Thus, this approach is also not suited for a trust-centered view.

A different approach to model distributed ledgers with BPMN collaboration diagrams is introducing a pool that represents the distributed ledger [165]. This interpretation bends the definition of a pool. In BPMN collaboration diagrams, pools illustrate separate organizational entities such as companies. Within these pools, different lanes present separate organizational units within an organization, e.g., a department within a company. While a distributed ledger is a data structure and not an organization, there are arguments to model it as a pool within a collaboration. DLTs draw their trust-enhancing capabilities not from the data structure but from their governance setup. Transactions, their validation, and consensus on their inclusions are the trust-enhancing features in a process.

A distributed ledger is democratically maintained by a set of organizations. In such a consortium, the consortium members maintain consensus over the current state of particular pieces of information. They validate transactions and agree on the state changes and their order. The consortium members maintaining the ledger may be the process collaborators that want to utilize it for its trust-enhancing capabilities. For public permissionless DLTs, the same principle holds. In that case, the consortium is not closed. Any entity can join the consortium and participate in the validation of transactions and maintenance of consensus. Thus, the consortium members create semantically a meta-organization that controls the ledger and the consensus. Subsequently, this consortium can be modeled as a pool in a collaboration diagram.

This section describes a semantic extension of BPMN collaborations that allows modeling DLTs following the described ideas (BPMN-DLT). BPMN-DLT illustrates the interactions of the collaborators with the ledger. Figure 3.18 depicts how transactions modifying state changes can be modeled. In BPMN terminology, these transactions can be interpreted as a message exchange from an organization to the ledger. The peer is not a pool in the process model. In BPMN-DLT, the technical level is hidden behind the layer of abstraction from a set of organizations maintaining the ledger. Also, the execution of consensus itself is a different level of abstraction. The message exchange may trigger the execution of activities. Parameters of the transaction are associated with a data

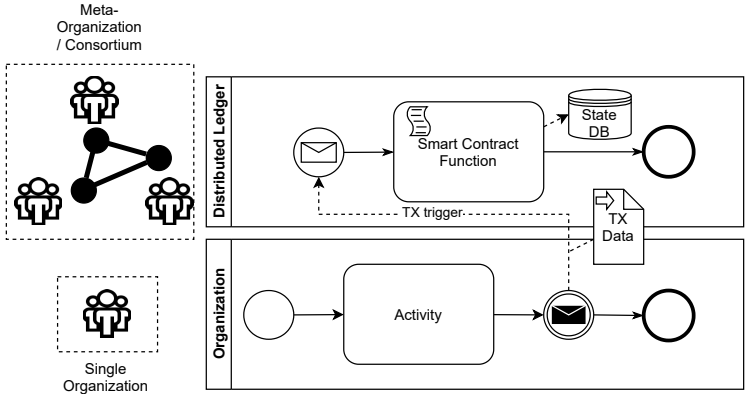

Figure 3.18: Modeling DLTs in a process: Transactions inflict state changes (write action).

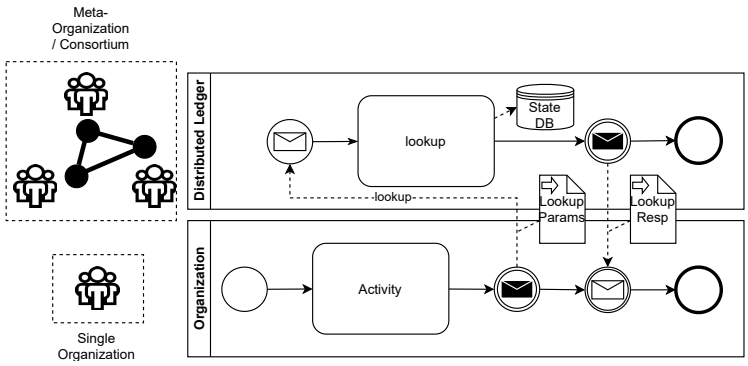

Figure 3.19: Modeling DLTs in a process: Looking up state information on-chain.

object. Technically, this is implemented by sending a transaction to a peer of the ledger. In DLT terminology, executing units of work can be interpreted as the execution of smart contracts. Therefore, BPMN-DLT reuses the script task BPMN marker. If the marker is placed within the DLT pool, this marks a smart contract function. Storing the current change itself can be seen as an interaction with the state database of the ledger. BPMN-DLT uses the database element from the BPMN standard.

Figure 3.19 illustrates the lookup state information stored on a DLT. Technically this is lookup of the DLT peer in its local copy of the state database. In BPMN-DLT, the lookup parameters are associated with a data object to a message exchange.

The previously proposed BPMN-DLT is a semantic extension of the BPMN 2.0 standard. Syntactically, all objects are already part of BPMN. The following descriptions of trust patterns with distributed ledger utilize the BPMN-DLT extension for illustration. The utilization of a collaboration diagram illustrates activities that are executed by one organization without the influence of others explicitly, which is in contrast to the approach by Ladleif et al. [127]. On the other hand, the introduction of the DLT meta-organization makes the responsibility of smart contract execution, consensus, and integrity maintenance clear. This mitigates the implicit assumptions made by López-Pintado et al. The base layer of consensus primitives and peer-to-peer interactions are modeled implicitly. This circumstance reduces the number of elements needed to depict the ledger. The BPMN-DLT extension uses the pure BPMN collaboration diagram syntax. Thus, it is fully compatible with the MCBPM meta-model. Hence, the following sections utilize collaboration diagrams with BPMN-DLT extensions to illustrate the DLT trust-patterns.

3.6.3 Reduce Uncertainty

Reducing uncertainty in a process aims to increase the likelihood that the respective parts of the process are executed as expected. Hence, this proactive approach targets the execution itself. The following trust patterns aim to reduce uncertainty, mostly regarding the correct execution (integrity) of activities. The correct emission and persistence (non-repudiation) of events is a second major

principle. Finally, reducing the uncertainty of wrongly executed process flows (message and sequence flows) can also be mitigated.

Distributed Ledgers as a Tamper-proof Hashed Data Storage

method	uncertainty root	trust concern	process element
reduce uncertainty	software, resource	integrity	data objects

In collaborative business processes, different organizations utilize shared data. Thus, the integrity of data is a major trust concern. Some collaborators might want to alter or manipulate data to gain certain advantages. This might lead to anomalies and malicious behavior for other process participants consuming the corrupted data. Verifying the integrity of data and trace its provenance is a challenge for process collaborators. Still, they have to rely on the provided data. Distributed ledgers can mitigate this uncertainty by storing data hashes to ensure data integrity. In terms of the TRUB model, the uncertainty concerning the integrity trust concern originates in the data. In a process model, this can be related to data inputs and outputs.

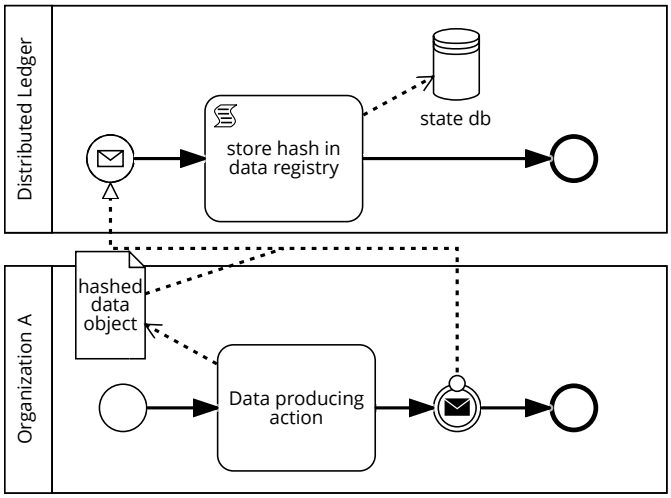

Figure 3.20: Process diagram for the distributed ledgers as a tamper-proof hashed data storage pattern.

The tamper-proof hashed data storage pattern works as illustrated in Figure 3.20. One organization produces a data object. The organization computes the hash of the data object. Afterward, the organization creates a transaction with the hash of the data object as a parameter. The organization submits the transaction to the distributed ledger. A smart contract stores the hash in the state of a registry contract. In that way, it is possible to group different hashed data objects. After some time, the transaction is included in the consensus, and the state change is finalized. In that way, a timestamp can be associated with the data object hash.

Data consumers can verify the integrity of the data object when they receive it. This is done by computing the hash of the received data object and a lookup of the

hash stored in the distributed ledgers. If the hashes do not match, data has been changed. Based on this principle, many different applications have utilized the blockchain as a tamper-proof hashed data storage. Examples include tracking of data provenance in Cloud Computing [166] or IoT-based collaborations [167]. The verification of data integrity needs to be executed within the activity that consumes the data. Therefore, all interfaces to the ledger need to be available to the activity.

The hashed data storage pattern enables a data-consuming organization to verify data integrity. However, how the information is processed cannot be restricted with this pattern.

In the running example shown in Figure 3.9 the sender creates SLAs. SLAs are guidelines for the carrier and his delivery services. Legally, SLAs are also the binding condition to be qualified for reimbursement. From the carrier's point of view, it is important that SLAs do not change during delivery. The sender could change the SLAs while the carrier is already delivering the parcel to circumvent paying for the service and commit fraud. Applying the hashed data storage pattern enables the carrier to prove the initial SLAs that the sender communicated. This is done by showing the local copy of the SLAs, computing the hash, and comparing it with the timestamped hash in the ledger's state. In the running example, it is possible to apply the same principle for a situation where the carrier wants to commit fraud and manipulate the SLAs.

Distributed Ledgers as a Transparent Process Event Log

method	uncertainty root	trust concern	process element
reduce uncertainty	organization	non-repudiation	event

Events represent reaching a certain state of interest in a process. In collaborations, events like failures during an activity execution might cause different workflows in the following process. An example of this situation is the error event. Error events might trigger error mitigation workflows in a process. Thus, it is essential that an organization cannot deny the occurrence of an event. They might want to do so as a fraudulent action to avoid compensation claims for failures in their domain of responsibility. In terms of trust-aware business processes, the organization causes a trust concern regarding the non-repudiation of an event occurrence.

A distributed ledger can store hashed data related to an event in an immutable fashion [168]. When an event occurs in a process, the responsible organization creates event details. These details are a short description of the event. The organization hashes the event details and creates a new transaction. The transaction includes the event details hash as a parameter. In the distributed ledger, the hash is stored in a registry in the state database. Afterward, the organization can communicate the clear text event details to the other collaborators in a process. In that way, it is possible to verify the integrity of the event occurrence. This works in the same way as the hashed data store pattern. An organization in possession of the clear event details object can compute the hash and compare it

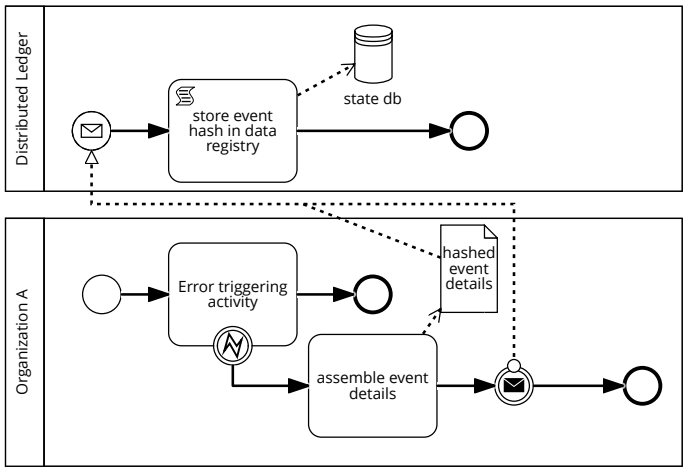

Figure 3.21: Process diagram for the distributed ledgers as a transparent process event log pattern.

with the hash in the distributed ledger's current state. Implicitly, the inclusion of the transaction with the event hash is timestamped. Thus, every event has a clear time of the occurrence that cannot be denied.

This pattern suffers similar limitations as the hashed data store. The organization responsible for the occurrence of the event is also responsible for its emission. If the organization wants to decide to keep the occurrence of an event concealed and not submit a transaction to the distributed ledger, the trust pattern remains ineffective.

In the running example shown in Figure 3.9 the carrier is required to file an incident if the firework parcel exploded during the delivery. If the parcel exploded, the carrier does not receive any money because the SLA was violated. Thus, the carrier has an incentive to disclaim the event occurrence later to get compensated nonetheless. If the carrier logs the incident to the ledger, the carrier cannot deny the event occurrence. Hence, non-repudiation is achieved.

Distributed Ledger Process Engine

method	uncertainty root	trust concern	process element
reduce uncertainty	organization	integrity	gateway, sequence flow, message flow

The correct control flow, message flow, and coordination between different collaborators are essential for executing an inter-organizational process. Thus, the integrity of the correct process flow is a major trust concern. Business process engines are commonly utilized to ensure the orchestration of a business process instance according to its defining model [37]. Traditional business process engines are meant to orchestrate the activities within one organization. Thus, in collaborations without a single centralized party in charge, coordination of different subprocesses carried out by different collaborators poses a significant trust issue.

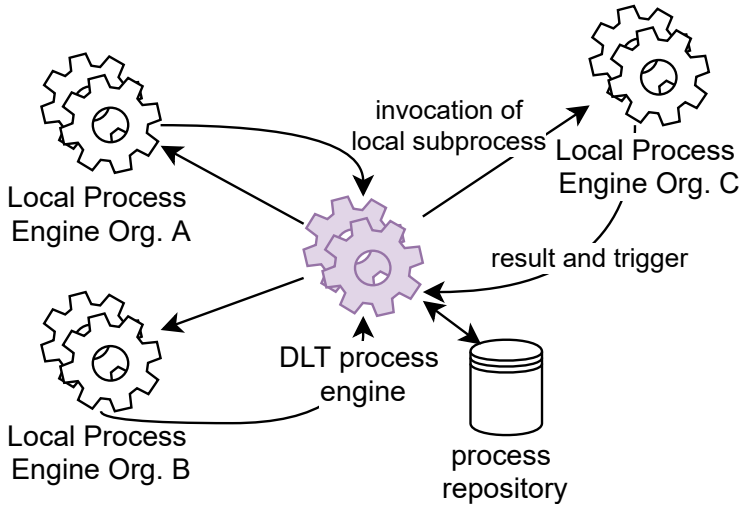

Figure 3.22: Conceptual illustration for the distributed ledger process engines.

Distributed ledger process engines store business process models in the state database of the ledger [130, 131, 136, 135]. A smart contract provides access to the process models. The process collaborators have access to the smart contract. They can verify the model's correctness at any point. Another smart contract implements a decentralized process execution engine. The engine creates a new instance of the business process as a new smart contract. The process flow is encoded in the business logic of the contract. Interfaces to local process engines can be attached to the DLT process engine using the observer pattern [169]. One can interpret this mechanism as invoking an external service for which a certain collaborator is responsible. The execution engine can also execute automated script tasks in smart contracts. Automated script task execution is further described in the smart contract activities trust pattern. Additionally, this pattern can also be seen as a way of software integration between different organizations [116].

Distributed ledger process engines cannot execute time-based sequences. In most DLTs, only a relative notion of time exists. In blockchains, for example, the notion of time is defined regarding the relative order of blocks. Concrete measurements of hours or minutes need to be injected through an oracle [120]. From a design perspective, smart contracts are meant to be triggered through transactions and not have a self-containing time observation functionality.

The process model from the running example in Figure 3.9 can be deployed to a distributed ledger business process engine. This engine can create new instances of the delivery process. Interfaces to the subsystems of the process collaborators are needed. The organization executes their local subprocesses in their realm of responsibility. One example is the "deliver parcel" activity. This is a manual task and hence cannot be executed in a smart contract. The system of the carrier needs to implement an interface to detect when the process engine triggers the execution of the parcel delivery task.

Smart Contract Activities

method	uncertainty root	trust concern	process element
reduce uncertainty	resource, software	integrity, availability	activity

Different organizations execute separate activities in a collaborative process. From an outside perspective, the execution of activities carried out by one organization can be seen as a "black box" for other organizations. Collaborators cannot verify the correct execution of the activities of other collaborators. It is also not traceable for them if the resources needed to execute the activity timely are available. The highly-available and transparent computing environment of distributed ledgers can mitigate this issue.

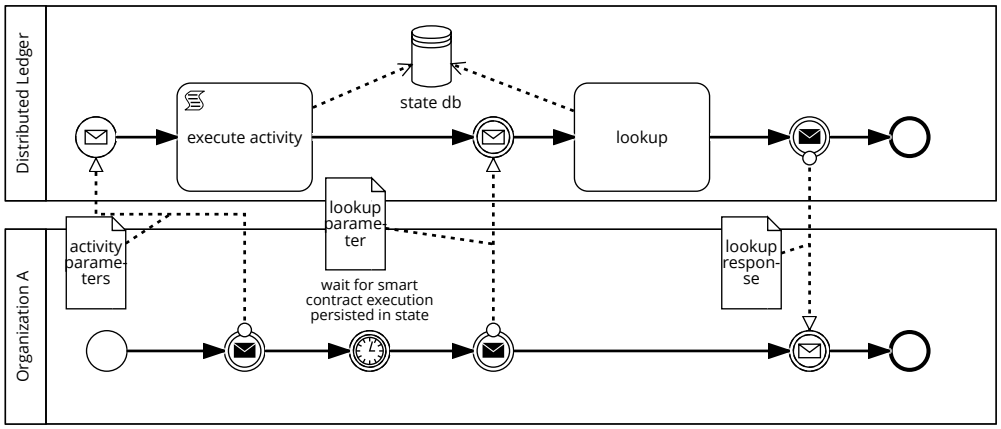

Figure 3.23: Process diagram for smart contract activities.

This trust pattern encodes the business logic of an activity in a smart contract. The smart contract is deployed to the distributed ledger. The execution of the smart contract is either triggered by the responsible collaborator or by another smart contract. Since a decentralized business process engine is implemented as a smart contract, it can also trigger the execution of a smart contract activity. A distributed ledger process engine triggering a smart contract activity can be interpreted as a ledger-based script task [130]. Hence, the distributed ledger process engine focuses on the correct orchestration of the whole process. Smart contract activities focus on the proper execution of one particular activity. Monitoring the results of a smart contract activity requires the invoking organization to trigger a lookup on the distributed ledger's current state. Smart contract activities ensure the integrity of the execution of the task. All organizations with access to the ledger can audit the code of the activity. Furthermore, availability is improved. The members of the blockchain consortium validate transactions and maintain consensus. Thus, a distributed ledger is a highly-available computing environment for smart contracts.

The distributed ledger execution environment limits the expressiveness of smart contract activities. Time-based business logic can hardly be encoded. Furthermore, data inputs to the smart contract activity can only be provided through input parameters. Other data needs to be available in the state database. This may lead to privacy issues. In addition, there is a limit on the complexity of

smart contracts. Also, resource consumption is often bound by a gas limit. This leads to the circumstance that smart contract activities are often limited to the execution of simple and short tasks.

The "create invoice" task of the running example can be automated. The invoice needs to reflect the SLAs and the way from start to finish. A smart contract can execute the price calculation.

Distributed Ledgers and Trusted Execution Environments for Activity Execution

method	uncertainty root	trust concern	process element
reduce uncertainty	resource, software	integrity, availability, confidentiality	activity

Executing activities in smart contracts often has limitations. In public distributed ledgers, the complexity of the logic is capped (e.g., with gas costs [16]). The execution of a smart contract that inflicts state changes needs to be traceable for the maintainers of the distributed ledger. This transparency leads to concerns about the confidentiality of the activity execution. Furthermore, also the input data to a smart contract activity needs to be available.

Distributed ledgers with smart contract execution in trusted execution environments can mitigate these privacy issues. Trusted execution environments (TEEs), such as Intel SGX [170] or ARM Trustzone [171], provide a hardware-based approach for trusted computation. A TEE is a secure enclave that resides within a physically shielded area of a processor. The host OS can enable interaction with the TEE from the outside but has physically no access to the inside of the TEE. Distributed ledgers can be used as a backbone to store job requests and execution certifications of TEEs. Examples for distributed ledgers and DLTs for trusted computing include iExec [172] and Ekiden [173].

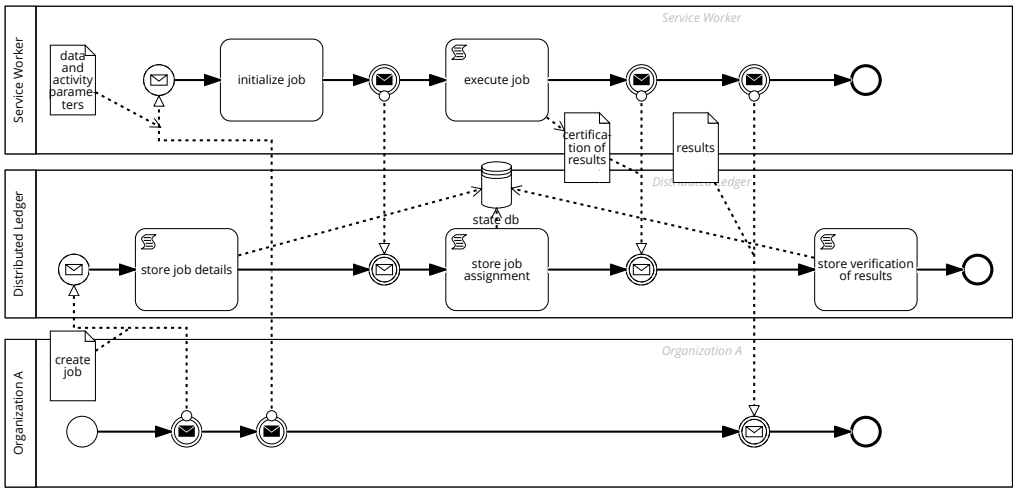

Figure 3.24: Process diagram for smart contract activities in TEEs.

Figure 3.24 sketches the high-level functionality of an activity execution in TEEs with DLTs. In a collaborative business process with TEEs, service workers

	Data	Activity	Event	Gateway	Sequence Flow	Message Flow
Integrity	HS	SCA, TEE	TEL	DLTPE, SCA	DLTPE	DLTPE
Confidentiality		TEE				
Availability		SCA	SCA	SCA	SCA	SCA
Non-rep.		TEL	TEL	TEL	TEL	TEL
Performance						
Resilience						

Table 3.6: Impact potential of distributed ledger to mitigate process related uncertainties: HS - Hash Storage, TEL - Transparent Event Log, DLTPE - DLT Process Engine, SCA - Smart Contract Activities, TEE - Distributed Ledger and TEEs

are organizations that provide their computing infrastructure as a service to other organizations. The hardware of the service worker includes a TEE. Service workers may be allocated and reimbursed for the execution of computing jobs. Therefore, the job details are stored in the state database of the distributed ledger. The input data and parameters needed for the activity are directly transferred into the TEE. Thus, the data is never available unencrypted on a distributed ledger. This prevents privacy issues regarding the data consumed in the activity. Before starting the execution of the activity, the service worker confirms its eligibility. Afterward, the job is executed in the TEE. A hashed version of the output of the activity is stored on the ledger once the job is finished. Additionally, also a certification of the job execution is stored in the ledger's state database. The clear text results are directly transferred from the TEE to the organization.

Using TEEs to execute activities is a technically challenging endeavor. The process above sketched a simplified workflow. Implementations, as in the iExec framework, involve complex encryption, data transfers, and attestation of the execution of code [172].

In the running example, a TEE may be used for the execution of the create invoice task. This would ensure that the invoice's price is calculated. Furthermore, nobody else than the organization in charge of the activity needs to know its business logic. Thus, integrity is ensured, while confidentiality is not challenged.

Table 3.6 shows a summary of the trust-enhancing capabilities of the discussed trust patterns to reduce uncertainty.

Non-DLT patterns to reduce Uncertainty Apart from distributed ledger trust patterns to reduce uncertainty, non-DLT patterns exist as well. This paragraph provides a non-exhaustive list of trust patterns that may be suited to mitigate trust issues in certain situation.

- **Increasing human resources for manual tasks**: In situations where a human resource is in charge of executing a manual task autonomously, it is possible to add another resource to ensure the integrity of the activity execution. This may be a supervisor or a co-worker to execute the activity jointly. This pattern is only applicable to manual tasks. Its trust-enhancing

capabilities highly depend on the trust tolerance profile of the involved persona.

- **Encryption**: Encrypting data mitigates the confidentiality trust concern to a certain extent. This pattern is limited in the sense that encryption can only increase the trustworthiness if the data processing organization itself is trusted.
- **Digital Certificates**: Signing certain data sets with a personal signature leads increased non-repudiation of data related to activities.

3.6.4 Reduce Vulnerability

Reducing uncertainty is a proactive approach to increase the trustworthiness of a process. However, it is not always possible to proactively prevent undesired effects in a process from happening. This is due to the impossibility to anticipate all exceptions that may happen, relying on human judgment (e.g., case-based decision making), or external factors. In such situations, reducing vulnerability is a different approach to increase the trustworthiness of a business process. This mitigation strategy aims to mitigate the *impact* to the outcome of the process once one or more uncertainties manifest. Rosemann states there can be different types of process vulnerabilities [9]. Time vulnerability might lead to increased process costs. For example, transportation companies, e.g., train or bus companies, might need to compensate their customers when their transportation process takes too long. This poses a time vulnerability. Product vulnerability describes the costs that result from an unsatisfactory quality of a product or service. For instance, in Germany, every customer buying products on e-commerce stores has the legal right to return the product within 14 days of arrival. Reasons do not need to be stated. In most cases, such returns are due to dissatisfaction with the product quality or from a different expectation of the product. This leads to increased costs for return shipment and handling. In general, vulnerability describes an increased cost due to a process not having the expected outcome. Thus, reducing process vulnerability is a reactive approach.

DLTs to reduce Vulnerability Distributed ledgers draw their trust-enhancing capabilities primarily from their inclusion in the execution of a process. Reducing vulnerability is focused on mitigating the impacts once something undesired occurred. Thus, DLTs are not suited to reduce vulnerability.

Non-DLT patterns to reduce Vulnerability Many vulnerability mitigation patterns can be used in a process as a reactive approach.

- **Re-execution of process parts**: For some cases, it may be possible to re-execute the part of a process that has not been executed as desired. For instance, in the process of producing a complex electric engine, the engine consists of different components. If the production subprocess of one of the components was not successful, this subprocess could be re-executed.

Thus, the defective component does not cause the whole engine to be faulty. This decreases vulnerability.

- **Process alternatives**: The unavailability of resources needed in the process may have severe consequences. If human resources required for the execution of a manual task are not available, the process might be delayed. In case of software or server outages, the same may hold. One of the most simple ways to mitigate these resources is to implement alternatives if some parts of the process have not been executed as desired. This may be a replacement employee in case another one gets sick or a backup server for outages.

3.6.5 Build Confidence

Reducing uncertainty and vulnerability are direct approaches to mitigate trust issues in a process. Building confidence is an approach to increase the *perceived* trustworthiness of a process. Therefore, this strategy adds supplementary sources of trust to the process. Sources of trust are usually independent of the process itself. Distributed ledgers enable higher level trust management systems or meta-process patterns to build confidence in the process. The following sections provide an overview of decentralized reputation systems and decentralization as a meta-pattern.

Decentralized Reputation Systems

method	uncertainty root	trust concern	process element
build confidence	organization	any	any

Reputation systems are a well-established approach for trust management in online commerce. These systems enable an assessment of the trustworthiness of a potential business partner by analyzing the experiences of others. Thus, they are an approach to build confidence in a process. Technically, reputation systems store the reputation *claim* that a reputation *source* raises regarding a specific reputation *target* [34]. Reputation systems have already seen their first implementations in the early ages of e-commerce, for example, for the online auctioning platform eBay [174]. Traditionally, reputation systems are centralized. In centralized reputation systems, a single authority (e.g., eBay for its marketplace) stores all reputation statements and performs the reputation aggregation for the user. This requires the user to trust the centralized authority since the authority can technically manipulate the stored reputation data.

Distributed ledgers may be utilized to implement fully decentralized reputation systems. In a decentralized reputation system, the reputation claims are stored transparently on a distributed ledger. Thus, the maintainers of the ledger ensure the integrity of the reputation statements. Regarding a process, reputation systems are external sources of trust that build confidence. In the current literature on decentralized applications, a large set of distinct approaches for application domains like academic reputation, tourism, or industrial IoT [175] is present. Yet, until today no universal decentralized reputation system that can be easily integrated in general applications has been proposed.

Decentralized reputation systems face many of the same security challenges, including white-washing or bad-mouthing attacks. Bellini et al. [175] provide a comprehensive survey of currently existing decentralized reputation systems and their application areas.

In the running example business process, the sender can use a reputation system to assess which carrier to trust for the transport of sensitive fireworks. The reputation statements need to be regulated and can be collected after the payment.

Distributed Ledgers for Decentralized Business Processes

method	uncertainty root	trust concern	process element
build confidence	organization	any	any

In collaborative business processes, different organizations execute private subprocesses autonomously. The subprocess executed by one organization is outside of the realm of influence of the other organizations. In case one organization is responsible for a particularly large fraction of the overall process, this indicates a significant dependency on the organization. Trust Mining's swimlane uncertainty balance (LUB) may detect such a situation. E-Commerce platforms are examples where such a large dependency on a single organization is present. They act as an intermediary between a buyer and a seller. During the acquisition and purchasing process, the platform is responsible for large parts of the overall process. For example, Amazon executes marketing services on behalf of the seller, delivers the items to the buyer, and processes the payment. This poses an accumulation of a large number of uncertainties at a single organization in the collaborative process. Decentralization, i.e., the distribution of different subprocesses to separate organizations, can counteract the dependency balance.

Distributed ledgers can be used as a tool to enable a better decentralization of a process. DLTs can connect subprocesses of different organizations. As such a software connector, DLTs provide an interface for message flows to different organizations [116]. They provide trust regarding the non-repudiation and integrity of messages. Furthermore, crypto-economic mechanisms can be utilized as a second way to enable better decentralization, for instance, through incentives.

The decentralization trust pattern does not reduce uncertainty. It can split the existing uncertainties in a more balanced way between organizations. The feasibility of decentralization strongly depends on the use case.

In the running example, the carrier is responsible for a large portion of the process. For instance, in the "deliver parcel" activity, the carrier causes uncertainty. It is essential for the other process stakeholders that an incident is reported correctly. One way to decentralize this process is to enable a different organization to monitor the parcel. IoT sensors can be utilized to observe the parcel conditions, similar to the approach presented in [176].

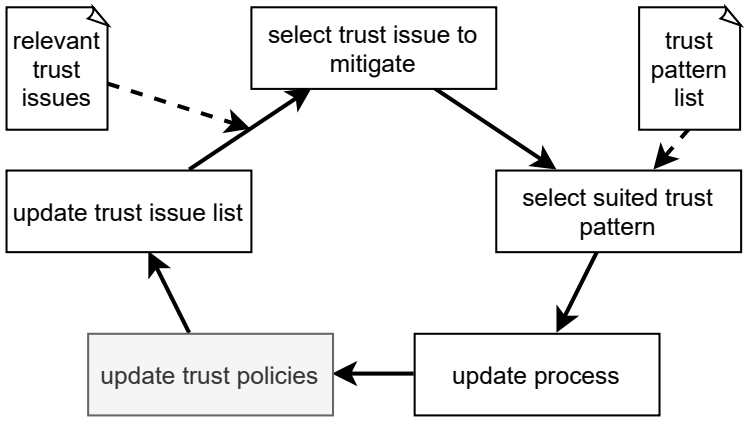

Figure 3.25: The five step approach to mitigate trust issues with trust pattern.

Non-DLT patterns to build Confidence Trust patterns that build confidence are not necessarily based on technological innovation. Traditional means to build confidence include legal frameworks or social relationships.

- **Legal Requirements**: Regulations that enforce certain mechanisms can act as a matter to increase confidence in a process. For example, the General Data Protection Regulation (GDPR) is a legal framework that defines how personal data can be processed [177]. Companies with business in the EU have to adhere to it for all their processes. Technically, laws like the GDPR do not restrict any processing. It incentives companies to adhere to it with fines.
- **Contractual Agreements**: Similar to legal regulations, contractual agreements define on a legal basis the terms of the process. For multi-sided platforms, users always need to accept the terms and conditions. Terms and conditions are a binding contract that can be legally enforced.
- **Social Relationships**: Relationships are another way to increase the perceived trustworthiness of a process. When process collaborators have been business partners for a longer period, then they perceive uncertainties differently compared to unknown partners [178].

The list of non-DLT trust patterns to build confidence is non-exhaustive and may be extended.

3.6.6 Applying Trust Patterns

Mitigating trust concerns with trust patterns is an iterative five-step approach, as illustrated in Figure 3.25. The mitigation process starts by selecting a relevant trust issue to mitigate. The trust issue can be characterized with the TRUB model regarding the relevant uncertainty, process element, and root of uncertainty. The next step selects a suited trust pattern to mitigate the trust issue. The classification of the trust patterns and trust issues are overlapping. Thus, the matching follows the previously discussed concept in Figure 3.16. Afterward, the process model is adjusted according to the trust pattern. In some cases, it

may be necessary to extend the trust policies. An extension is needed in cases where the trust pattern introduces new trusted elements that were previously not reflected in the trust policies. Finally, the list of relevant trust issues can be updated, and the next mitigation cycle can begin.

The next paragraphs discuss the five steps in detail and provide an example for illustration.

Select Trust Issues to Mitigate The first step of the TAPE method identifies a list of relevant trust issues. When a process engineer wants to mitigate these trust issues, a prioritization is necessary. The prioritization depends on the overall goal and motivation of the trust persona.

In the running example, the process engineer consults a few large customers of a logistics company. The engineer finds out that the integrity of the SLA definitions is one of the most critical points to mitigate for the customers. Thus, the engineer decides to mitigate the trust issue regarding the SLA document in the processes caused by human resources and software as a first priority.

Select suited Trust Pattern After the trust issue to mitigate has been selected, the process engineer needs to identify the trust pattern for mitigation. Depending on the trust issue, there may be several candidates applicable. In this case, the process engineer has to analyze the limitations and context factors to select the ideal trust pattern. The trust pattern list serves as an input to this step. The list is classified according to the taxonomy and matches the TRUB model. Thus, selecting potential trust patterns is a simple comparison of the trust concern, uncertainty root and process element type.

In the running example, the process engineer uses the list of trust patterns as introduced in previously. The engineer selected the distributed ledger hashed data storage pattern as a way to ensure the integrity of the SLA definition.

Update Process Modifications on the As-Is process with the selected trust pattern lead to a new version of the process. This step may include adding or deleting process elements or modifying the process flow.

The incorporation of the trust pattern to the running example is shown in Figure 3.26. The sender stores a hash of the SLA definition in the state database of a distributed ledger. For later conflict resolution actions this provides a tamper-proof record of SLAs.

Update Trust Policies In the case that the trust pattern introduces new roles or elements to the process, it may be required to update trust policies.

In the running example, the trust pattern added a distributed ledger to the process. Thus, it is needed to express that the sender, carrier, and receiver all trust the distributed ledger consortium of maintainers. Table 3.7 sketches the update of the trust tolerance profile of the sender trust persona. A new policy shows that the distributed ledger is trusted for all process elements and trust concerns. Equivalently, the trust tolerance profiles of carrier and receiver may also require updates.

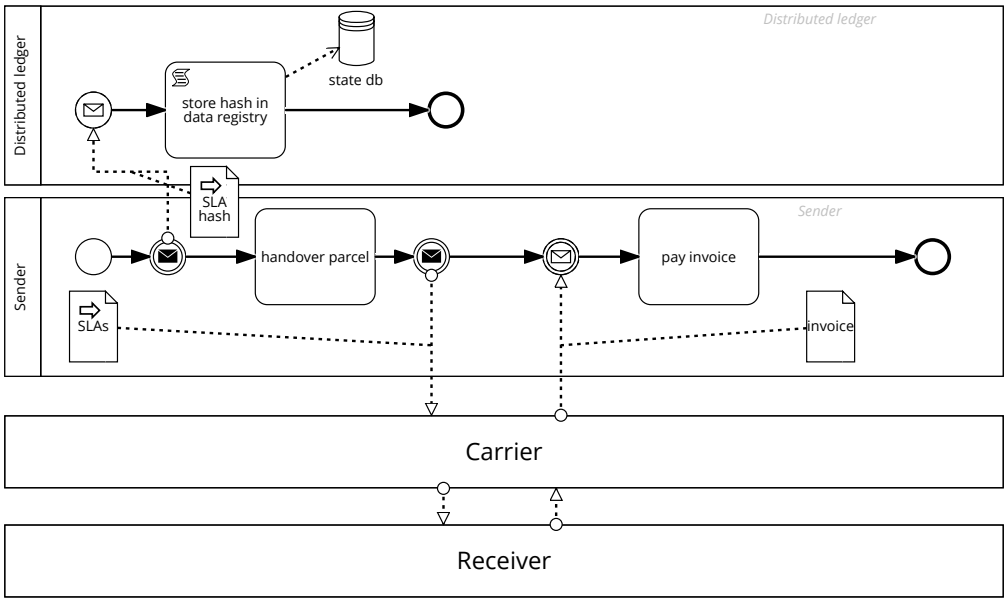

Figure 3.26: Process model of the running example with mitigation. The DLT hashed data store pattern introduces a new pool in the process. The roles of the carrier and receiver are collapsed for simplicity.

Trust Persona	sender	
Trust Entity	**Trust Subject**	
	Process Element	**Trust Concern**
…	…	…
distributed ledger	all	all

Table 3.7: Adding a trusted ledger to the process requires updates of the trust policies.

Update Trust Issue List With the mitigation in place, the list of relevant trust issues can be updated. In the TAPE method, this means re-executing Trust Mining. The re-execution yields a different set of relevant trust issues. These trust issues are subject to the subsequent cycle of trust issue mitigation.

This section described one iteration of the application of trust patterns to the process. A full example with a supply chain example can be found in a case study in Section 5.4.

3.7 Implementing Trust-aware Business Processes

The implementation of a trust-improved business process utilizes the process model from the previous step as a reference. In general, the implementation step includes system selection, implementation, and testing of the trust-improved

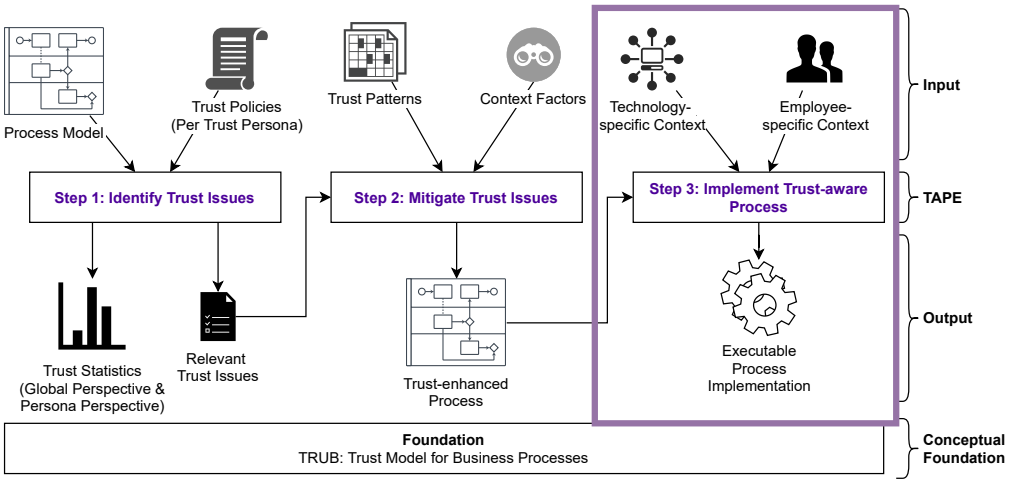

Figure 3.27: Implementing a trust-aware business processes in the context of TAPE.

process. This section describes the unique properties of distributed ledgers that need to be considered during the implementation phase.

From an information systems perspective, distributed ledgers store state information of data objects. Only transactions can alter the current state. The ledger's maintenance consortium's network peers validate transactions, order them linearly, and link them cryptographically together. This linkage makes the transactions immutable. Thus, the integrity of the stored information can be guaranteed as long as a majority of the network behaves truthfully. The consensus protocol ensures that no inconsistencies between the peers exist. Thus, the distributed ledger maintains a synchronized state across all the peers. Every organization with access to the state database can verify the validity of transactions and the current state of particular pieces of information. Hence, distributed ledgers provide the following information-centric properties to a process:

- *Immutability of past transactions*: Once a transaction was included in the ledger, it (practically) cannot be altered.
- *Consensus in linear log*: All peers maintain consensus on a synchronized state of information stored in the ledger and changes are ordered linearly.
- *Transparency*: Network peers can verify the validity of transactions.

Distributed ledgers draw their trust-enhancing capabilities from these three properties. However, these properties imply practical challenges. The challenges are mostly centered on non-functional characteristics. They cannot be sufficiently assessed on the abstraction layer of a business process model. Thus, the process engineers and system architects need to consider these challenges during the implementation phase:

- *Finality*: Transactions submitted for validation to distributed ledgers are not instantly synchronized to all peers. The peers need to reach consensus first.

- *Governance*: Consensus on the ledger's state is maintained by different peers. When they gain a large enough influence in the network, they can exploit the network to gain benefits.
- *Privacy*: The ledger's state is usually visible to all peers, so that the state is verifiable.

The following sections discuss these main challenges inherent to distributed ledgers. Furthermore, the following sections establish methods for addressing the finality, governance and privacy challenges when implementing trust-aware business processes with distributed ledger technologies.

3.7.1 Finality

In a distributed ledger consortium, different peers from separate organizations maintain consensus on the ledger's state. Most consensus protocols incorporate multiple phases. However, they have common characteristics that can be regarded as *meta-phases*. First, a new transaction is submitted to the network to get validated and reflect the included changes in a future state update. Next, the transaction gets (depending on the consensus protocol) validated by one or several peers. After the initial validation, the transaction needs to get ordered and propagated to the other peers. The other peers in the network also validate the transaction. Afterward, they decide whether to include the transaction in their version of the ledger's state database. A transaction can be considered practically final after the majority of network peers accepted it.

The time between the first submission of a transaction to the network until it reaches practical finality depends strongly on the consensus protocol. Some protocols enable near-instant finality, while others require several hours. Different types of consensus protocols imply different properties regarding finality. Blockchain-based distributed ledgers often provide *probabilistic finality*. One example of probabilistic finality is Bitcoin's Nakamoto consensus [15]. Network peers order transactions into different blocks. After assembling a new block, a peer sends it to the other peers for propagation. Different peers may have a different state stored at a point in time. The order of the state changes can be validated by tracing the blocks that are cryptographically linked. During the consensus, the longest chain wins. Thus, the transactions in older blocks sink deeper into the chain with every new block. This increases the probability that no fork will replace the current one that contains the transaction. In general, finality can never be reached to 100% in such protocols. In contrast to probabilistic finality, protocols that are based on practical byzantine fault tolerance (PBFT) often offer *absolute finality* [71]. Forks do not occur in such protocols as they do in the Nakamoto consensus. Thus, no "race" for the longest chain exists. The transactions are final once the needed number of validators approved them. *Economic finality* implies that older blocks can only be reverted with an ever-increasing economic expenditure. Examples for economic finality are Proof of Stake protocols [72].

When implementing trust-aware business processes with distributed ledgers, finality must be considered thoroughly in user interface design, software artifacts, and their backend implementation. Transactions to traditional centralized databases are nearly instant. Hence, the feedback loop from triggering a transaction in the user interface of an application to the backend database is negligibly small. Feedback loops are important for a user to display success or error messages. When a distributed ledger is involved in the process, this might take a significantly longer time. An application that uses a DLT needs to raise awareness that actions are not necessarily instantly final. This can be achieved in a synchronized or asynchronous way. The synchronized approach shows a blocking loading indication to the user while the triggered transaction awaits practical finality. For asynchronous solutions, it is necessary to sufficiently indicate to the user that the action has not reached finality yet. Highlighting non-finalized interactions with an icon can achieve this effect visually. The application also needs to be able to handle requests to a distributed ledger asynchronously. With DLTs, an additional set of errors can occur when compared to a centralized database. Logic needs to implement error-handling accordingly.

In the trust-enhanced process in Figure 3.26, a web interface lets the user define SLAs and store them in the backend. A spinner can indicate that the associated transaction has not reached finality yet.

3.7.2 Governance

Distributed ledgers replace centralized parties that are in control of shared data with a decentralized network. In this network, different peers from separate organizations control the information in the ledger jointly and maintain its integrity. Thus, DLT consortia are subject to different political choices that reflect how users can interact with the DLT system [179]. Governance decisions of the ledger imply different characteristics of the network regarding decentralization, censorship-resistance, and tamper-resistance.

The peers of a DLT network engage in a consensus with specific rules to ensure the state database's integrity. In case the majority of the network conspires to invalidate transactions from the past, a decentralized network does not provide benefits compared to a a centralized service provider. In the running example of the delivery process for dangerous goods, the SLAs are stored in the distributed ledger's state database. In case the carrier denies the existence of SLAs, the sender can use the transaction to the DLT as proof of the document's integrity. If a majority of the network peers conspires to invalidate the transaction, then the trust-enhancing capability of the ledger is not applicable anymore. Thus, trust in the network's governance is an essential factor to consider when implementing trust-aware collaborative processes with distributed ledger technologies.

A central governance decision to make when implementing trust-aware business collaboration with a distributed ledger is choosing a *permissioned* or *permissionless* ledger. In current literature, these terms are used differently. The following sections use the definitions as established in Chapter 2. In permissionless distributed ledgers, every entity with access to the network may participate in the consensus. They may gather submitted transactions, verify transactions,

and add them after successful verification and consensus to update the ledger's state. Two examples of permissionless ledgers are Bitcoin [15] and Ethereum (public main-chain) [16]. In contrast to permissionless ledgers, permissioned DLTs restrict the set of entities that participate in the consensus. Examples for permissioned DLTs include Ripple [57] or Monax [58].

The mere introduction of a permissioned network does not mitigate problems regarding tamper-resistance compared to a permissionless network. When deciding to implement a permissioned network, it is also necessary to assess thoroughly *who* the governing parties in the network are and what their interests are. The consortium needs to be trusted or incentivized not to form alliances and attack the network. For example, Ripple [57] is a consortium network that enables fast cross-country money transfers. Its main users, i.e., banks, govern it. The motivation to use a DLT is to improve the speed of international money transfers between different banks. Thus, they have an intrinsic interest in ensuring the integrity of the system. If any of the involved parties decided to attack the network, the system would not be usable. Thus, this would invalidate the motivation to use a DLT in the first place. It is important to analyze which incentives the peers in the consortium might have. This holds not only for maintaining the ledger's integrity and providing tamper-resistance but also for other non-functional properties. For instance, some peers might favor transactions submitted by certain participants over others. This could lead to different times to finality for transaction-submitting participants and may be seen as a form of discrimination.

Thus, assessing the governance requirements, risks, and potential consortia for a permissioned network is crucial in the implementation phase of trust-aware collaborative business processes. The right solution for the specific use case strongly depends on the collaboration setup and trust tolerance profiles regarding the network peers.

3.7.3 Privacy

For implementing trust-aware business processes, privacy is a one of the major challenges that needs consideration. In a distributed ledger, no past transaction record can be deleted. Other peers can at any point in time trace all past transactions in a ledger to validate the current state. This provides a way to maintain the integrity of data stored on the ledger. At the same point, this poses a massive privacy issue.

Several approaches to enhance privacy in distributed ledgers have been proposed [180]. The majority of them is focused on cryptocurrencies. The notion of DLTs as a general computation framework with smart contracts facilitated in collaborative business processes has not been explored prior to this work. Feng et al. [108] distinguish between two different types of privacy requirements in blockchains: identity privacy and transaction privacy. Identity privacy describes the linkability between the wallets used to emit transactions and the real world entities behind it. Transaction privacy means that the transactions' contents are only visible to actors who are semantically involved. From these privacy requirements, concrete privacy threats result. De-anonymization deals

with reconstructing the link between on-chain transactions and real-world entities. Therefore, different approaches based on network analysis [181] and address clustering have been proposed in different publications [107, 182]. Besides de-anonymization, techniques to analyze the semantics of blockchain transactions exist. Hence, transaction graph analysis [106] or process mining [124, 183] can be utilized. For more detailed information on distributed ledger analytics see [184].

From a subject perspective, work on privacy-enhancing technologies (PETs) mostly distinguishes between *transaction privacy* and *smart contract privacy*. Regarding transaction privacy, the most commonly studied privacy property is anonymity. On the other hand, smart contract privacy focuses mostly on hiding details of the smart contract's internal state. In the context of distributed ledger trust patterns for collaborative business processes, both types are of relevance. Every smart contract execution is triggered through a transaction from the outside. Thus, transaction privacy is implicitly relevant for all trust patterns as discussed in Section 3.6. In every pattern, where more as the mere logging of information on-chain is required, smart contracts are employed. Hence, smart contract privacy becomes relevant. Examples for that are distributed ledger business process engines or smart contract activities.

3.8 Summary

This chapter introduced concepts to analyze trust in collaborative business processes, mitigate trust issues with trust patterns, and implement trust-aware processes.

- Section 3.2 defined the field of trust-aware business process management.
- The TAPE method is a general three-step method for the analysis and implementation of trust-aware processes. Section 3.3 introduced a high-level overview of TAPE.
- Section 3.4 introduced the TRUB trust model. TRUB relates trust-related concepts to elements of business process models. It serves as the foundation of the TAPE method.
- TAPE's first step is the identification of trust issues in a business process. Section 3.5 introduced Trust Mining as a method to automatically analyze potential trust issues using business process models.
- The second step mitigates identified trust issues with trust patterns. Section 3.6 introduced distributed ledger trust patterns as tools to mitigate trust issues in a process. All trust patterns are classified regarding the TRUB model.
- The final step of TAPE aims to implement a trust-aware process. Section 3.7 discussed guidelines for the implementation and challenges specific to distributed ledgers in collaborative business processes.

This chapter discussed the novel concepts that this thesis proposes. The following chapters aim to implement support tools and evaluate the concepts.

4 Implementation

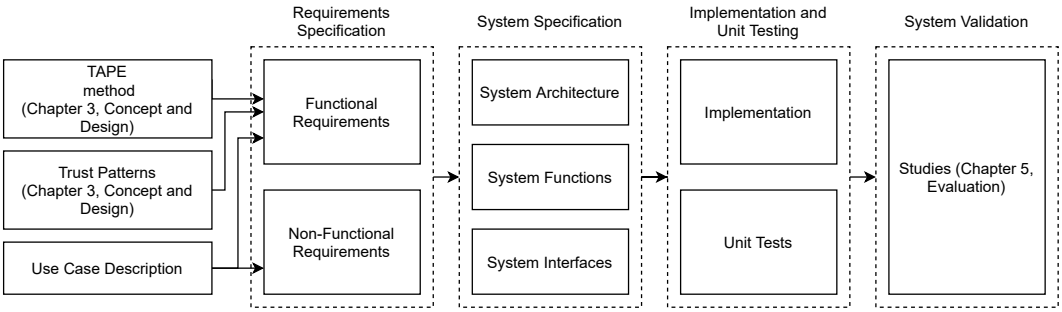

Figure 4.1: Structure of the implementation of TAPE with Trust Studio

TAPE is a complex method that is designed with automation in mind. Fully manual methods are susceptible to failures of the user to apply certain aspect of the method correctly. This may result in undesired outcomes. This chapter presents Trust Studio. Trust Studio is a tool to support process engineers and analysts in applying the TAPE method. Conceptionally, Trust Studio and the TAPE method are tightly connected. Trust Studio cannot exist without the TAPE method. Yet, it is possible that the TAPE method uses another tool to support its users in its application. Trust Studio is merely a reference implementation of a support tool for TAPE.

Conceptually, implementing Trust Studio follows the methodology depicted in Figure 4.1. The implementation follows the overall phases in software engineering [185]. The TAPE method, the set of identified trust patterns, and the use case descriptions constitute the inputs to the requirements specification phase. Functional requirements are derived from all of the three input artifacts. Non-functional requirements are primarily derived from the use case descriptions. The system specification describes the architecture, system functions, and interfaces to other systems or users based on the requirements specification. After these design steps, the system is implemented. The system validation is part of the evaluation in Chapter 5.

4.1 Use Cases

This work envisions two primary use cases for the TAPE method. The *descriptive* approach lets process stakeholders *comprehend* trust-related concepts in their process. This use case requires detailed insights into the relationships that are present and the issues that these imply. The *creative* approach also analyzes the process model but also aims to transform a process. Hence, the descriptive use

case's ultimate goal is trust assessment, while the the creative approach targets mitigating trust issues.

4.1.1 Descriptive Use Case

TAPE may be used in a purely descriptive manner. It enables process stakeholders to understand to which trust-related situations they commit. For instance, Alice wants to start a new e-commerce business. After an initial phase of market analysis, she selects a product to sell. Alice is very passionate about Kombucha. Kombucha is a fermented, lightly fizzy, sweetened black tea usually sold in mason jars or bottles. She has connections to a Japanese brewery that is capable of producing it industrially. Alice requires a logistics company to deliver her fermented black tea to her customers. She wants to serve customers worldwide. Alice believes that customer satisfaction is a critical factor in building a sustainable business. Yet, large parts of the value chain are out of her own direct control.

Thus, Alice decides to use TAPE's first step to discover potential trust issues and comprehend which relationships she needs to commit. The steps to mitigate trust issues and implement the process are at this point not relevant to her. With Trust Mining, she first needs to model the process formally. Alice decides to use the standard configuration parameters for uncertainty roots and trust concerns that Trust Mining defines. The metrics indicate an uncertainty imbalance regarding second-level carriers. Alice thinks deeper about the topic and realizes the situation. While she might directly relate to her first-level carriers, these might outsource certain parts of the logistics chain to other carriers. For example, in rural regions, major carriers often delegate delivery to local logistics companies.

Technically, the correct handling of documents and agreements causes uncertainties. Alice realizes how trust-intense the process is. When working with a carrier that does not handle SLAs well, a lot of conflict resolution may be required. Alice is looking for an e-commerce niche where she has minimal conflict resolution activities. TAPE and Trust Mining's insights make her considering starting an e-commerce business that does not require physical items and carriers to bring them to the customers. In this example, Alice purely utilized TAPE to understand what trust constellations she has to position herself in and made a business decision. This is the general outline of the descriptive use case.

4.1.2 Creative Use Case

TAPE can also be the starting point for the trust-aware process (re-)engineering. The process of trust-aware (re-)engineering starts by analyzing all relevant trust issues separately. Afterward, applying trust patterns modifies the process and might improve present trust issues.

For instance, Bob owns an e-commerce store where he sells luxurious vegan sneakers. Currently, his primary business is focused on central and northern Europe. Yet, he considers expanding his business to southern Europe. He has to find new partners that will deliver his packages in the new region. With his

current partners in central Europe, he has an excellent business relationship. Bob has been working with his current carriers for years and trusts them. However, he has no partners in the new region. Therefore, he wants to make his process as trustless as possible. Thus, Bob decides to use the TAPE method for improving his process regarding trust-related aspects.

Bob uses Trust Mining and analyzes the metrics. There, he sees that much uncertainty originates from the correct handling of documents in the process for SLAs. Bob's business targets high-value fashion for an environmentally conscious and wealthy customer. Thus it is important that the packaging of the sneakers looks pristine when it arrives at the customer's house. Bob decides to modify his process and mitigate the trust concern of a carrier denying the commitment to SLAs. Therefore, he introduces a digital agreement in his process. This agreement gets stored in Bob's cloud backend. A timestamped hash of the agreement is submitted to the Ethereum blockchain, following the DLT tamper-proof hash storage pattern as described in Section 3.6. After the implementation of the concept, it is harder to deny that the SLA exists. Hence, this is not a relevant trust issue for Bob anymore. Bob utilized all phases of the TAPE method in this creative use case to improve his process.

4.2 Requirements Specification

The two primary use cases serve as the base for Trust Studio's requirements specification. The following two sections discuss functional and non-functional requirements for Trust Studio. In general, the scope of Trust Studio is to support the first two steps of TAPE. The identification and mitigation phase can be executed without any access to their implementation. Implementing a trust-aware business process, on the other hand, requires technical and organizational access to the systems and environments in which the process will be executed.

4.2.1 Functional Requirements

Table 4.1 shows how the use cases can be related to system requirements and system functions. In both use cases, the user needs an interface to the system. To identify uncertainties in a process (TAPE Step 1), the user needs to describe the business process (R1). Therefore, Trust Studio needs to provide a BPMN modeler to the user (F1). The modeler can define a new process model or upload an existing BPMN model in XML-Format. Next, in Step 1, in both use cases, the user needs to get support in annotating the process model with uncertainty possibilities. This requires system function F2, the automated annotation of uncertainties. Automating this step leads to fewer issues in applying TAPE due to human error. In both use cases, the user might want to define trust personas (R3). Therefore, Trust Studio should provide an interface where the user can define trust personas and their respective trust policies. In the analytical part of TAPE, the user requires to see the trust metrics of the input process on a global level (R4) and on a persona level (R5). Therefore, Trust Studio needs functionalities to calculate the metrics (F4.1 and F5.1) and display them

Use Case	TAPE Step	System Requirements	System Function
1,2	1	R1 describe business process	F1 BPMN modeler
1,2	1	R2 annotate process model with uncertainty	F2 uncertainty annotation
1,2	1	R3 create trust personas	F3 persona creation
1,2	1	R4 visualize global metrics	F4.1 global trust metric calculation, F4.2 global trust metric visualization
1,2	1	R5 visualize persona metrics	F5.1 persona trust metric calculation, F5.2 persona trust metric visualization
1,2	1, 2	R6 visualize relevant trust issues	F6 list relevant trust issues
2	2	R7 update process model with trust pattern	F1 BPMN modeler
2	2	R8 update trust persona	F7 persona update
2	2	R9 update uncertainties	F2 uncertainty annotation
1,2	1	R10 visualize global metrics	F4.1 global trust metric calculation, F4.2 global trust metric visualization
1,2	1	R11 visualize persona metrics	F5.1 persona trust metric calculation, F5.2 persona trust metric visualization

Table 4.1: The table depicts the use cases and relates them to their steps in the TAPE method, requirements and envisioned system function. UC1: descriptive use case, UC2: creative use case.

appropriately (F4.2 and F5.2). Furthermore, both use cases require visualization of the relevant trust issues. Therefore, Trust Studio needs to display a list.

The creative use case (UC2) requires more functionality than the descriptive use case (UC1). Based on the insights in the first step of TAPE, the second step is also relevant in UC2. Thus, the user needs to apply trust patterns to the current process and modify it. Therefore, the system is required to update the process model with trust patterns (R7). Trust Studio reuses the BPMN modeler (F1) that can also modify an already existing process model. When the model is updated, an update of the uncertainties (R7) is also required. Therefore, Trust Studio can reuse the metric calculation functionality F2. In some cases mitigating trust issues introduces new elements to the process. For example when a distributed ledger is introduced to a process where none was present before, it is also required to specify the trust policies towards the peers that maintain the ledger. Thus, updating trust policies of certain trust personas is also required (R8). Functionality-wise, Trust Studio can reuse the existing trust policies and update them for a specific trust persona (F7). With this modification to the process, also the uncertainty annotation (R9) needs to be updated. The same holds for the visualizations (R10 and R11). The same functions used for the initial computation can be reused for the changed model (F2, F4.1, F4.2, F5.1, and F5.2).

System Function	Non-functional Requirements
F1 BPMN modeler	NFR1 sufficiently usable
F2 uncertainty annotation	NFR2 in less than 3 seconds for a usual BPMN diagram with 50 elements
F3 persona creation	NFR3 sufficiently usable
F4.1 global trust metric calculation	NFR4.1 in less than 3 seconds for a usual BPMN diagram with 50 elements
F4.2 global trust metric visualization	NFR4.2 sufficiently understanbale
F5.1 persona trust metric calculation	NFR5.1 in less than 3 seconds for a usual BPMN diagram with 50 elements
F5.2 persona trust metric visualization	NFR5.2 in less than 3 seconds for a usual BPMN diagram with 50 elements
F6 list relevant trust issues	NFR6 sufficiently understandable
F7 persona update	NFR7 sufficiently understandable

Table 4.2: Non-functional requirements for Trust Studio related to their system functions.

4.2.2 Non-Functional Requirements

Trust Studio aims to aid the users of TAPE to execute TAPE-related tasks. Therefore, also non-functional requirements need to be fulfilled.

Table 4.2 shows non-functional requirements for Trust Studio and their relationship to the corresponding system functions. Overall, the non-functional requirements originate from ease of use, usefulness, and execution speed. Trust Studio's parts that require direct input from the user (F1 and F3) should be easy to understand. The definition of easy-to-use is fuzzy and depends on the user. As a reference, a general user profile of a process engineer or analyst poses the target group of Trust Studio. These personas are typically familiar with web interfaces, uploading files, and typing information into HTML tables. All functions that require computation in Trust Studio (F2, F4.1, F4.2, F5.1, and F.2) need to be sufficiently fast. Calculating the metrics takes longer than three seconds, the user might lose patience with the application. All parts of Trust Studio that illustrate insights, such as the trust metric and trust issue list (F4.2, F5.2, and F6), need to be sufficiently understandable to the user. If the visualizations are too abstract, the user might refrain from using Trust Studio at all.

4.3 System Specification

From an architecture point of view, Trust Studio has a simple structure. The whole application is implemented as frontend web application that does not need any backend functionality. It has four basic function blocks, as illustrated in Figure 4.2. The configuration component (1) lets the user personalize Trust Studio based on the needs in the use case. For example, the uncertainty possibility list may be configured with this component. The modeling component (2) of Trust Studio lets the user define the process model to analyze. Furthermore, the component allows to upload already existing process models in

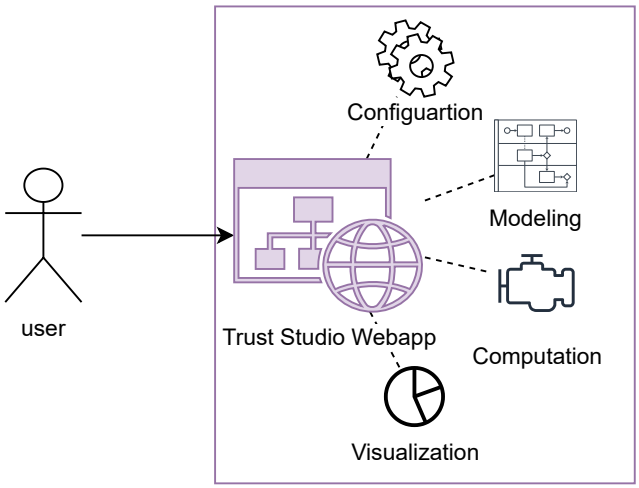

Figure 4.2: Schematic architecture of Trust Studio.

the XML-representation of BPMN. The computation component (3) automates the uncertainty annotation, relationship analysis, and trust metric calculation of TAPE. The visualization component (4) illustrates the outcomes of these calculations to the user.

From an operations point of view, Trust Studio does not execute programming logic on a server. For BPMN diagrams with less than 100 elements, the computing capabilities of modern web browsers are sufficient. Trust Studio can be deployed locally or on a web server for remote access. Authentication or authorization is not required, since all computing is executed in the user's web browser.

4.4 Implementation

Trust Studio implements the discussed functionality in a javascript-based web application utilizing the React.js framework[1]. Trust Studio has a modeling component, where a user can upload an existing BPMN file. Users can also edit their process models or define a new ones using the open-source BPMN.IO library[2].

The software adds the trust-related concepts as custom artifacts to the process model. Users can define the uncertainty possibility list in several ways. They can utilize a web form, select a predefined list, or upload a CSV-file of uncertainty possibilities. Users can define trust policies for different trust personas and generate metrics based on them.

The following sections discuss the user interface and interactions in detail.

Model Definition Before the software can execute any of the steps of TAPE, Trust Studio requires a process model. The user can upload an existing model through the web interface's upload button (see Figure 4.3 upper right corner). Therefore, Trust Studio requires the file to be in a BPMN 2.0 compatible XML encoding [38]. Alternatively, a user can also define the model directly in Trust

[1]https://reactjs.org
[2]https://github.com/bpmn-io

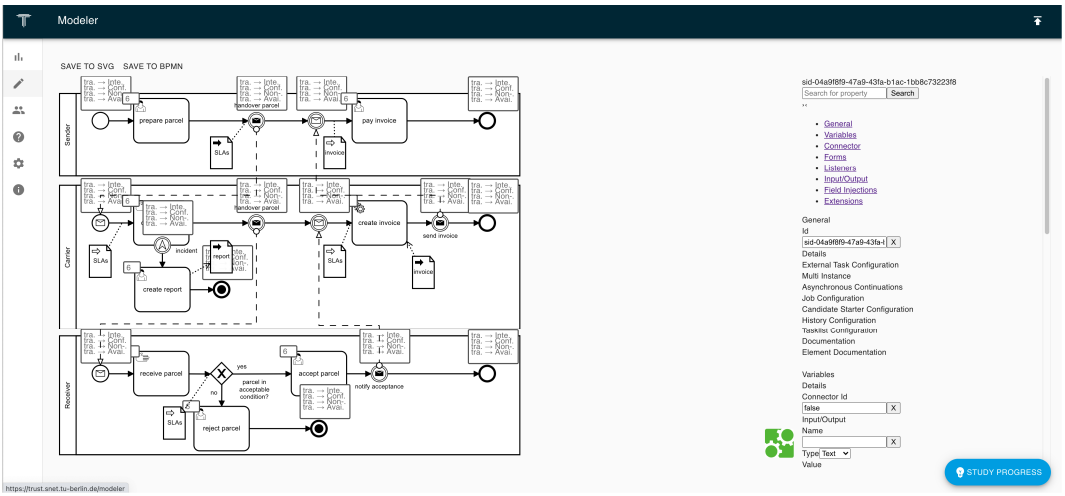

Figure 4.3: Trust Studio user interface: BPMN modeler.

Studio's modeling component. This modeling component uses the open-source BPMN.io library[3]. BPMN.io is developed and maintained by the developers of Camunda [186] and is compatible with React.js. When a user uploads a new process model, Trust Studio automatically annotates the uncertainties as depicted in Figure 4.3 (left). Every time the model changes, the annotation is updated, and the trust metrics are recomputed.

Trust Visualization The user can get an overview of the trust-related properties of the process model in the visualization dashboard as illustrated in Figure 4.4.

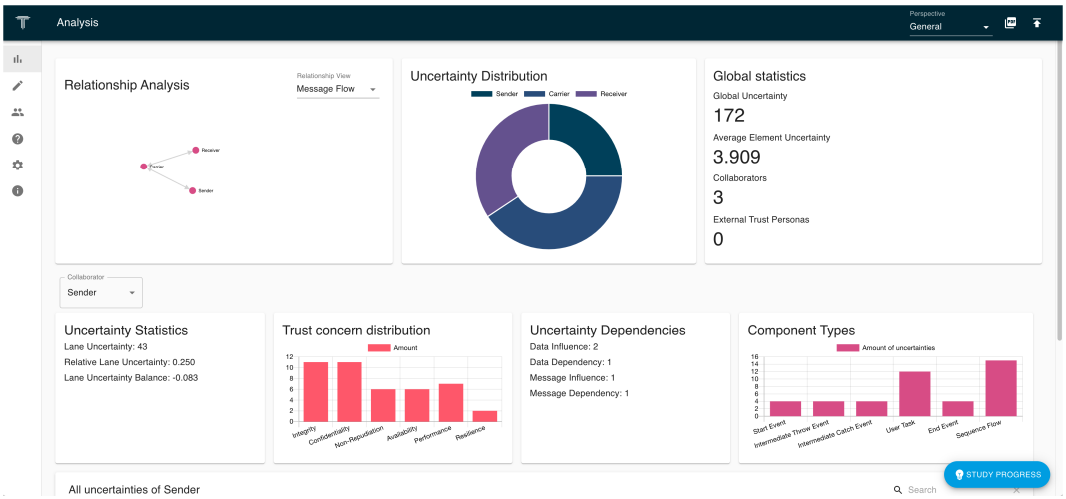

Figure 4.4: Trust Studio user interface: trust dashboard.

In the upper right corner, the dashboard shows a section related to relationship analysis. In this section, the user can decide to show the data dependency or message dependency graph. The graphs are built according to the definitions in Chapter 3. The upper center diagram shows the uncertainty distribution of the process. This donut chart defines its sections according to the average lane uncertainty (ALU) of the respective process collaborator. The right table shows

[3]https://github.com/bpmn-io

a textual description of the global uncertainty (GU), the average uncertainty per element, and the number of trust personas that a user modeled.

The bottom section of the dashboard depicts statistics related to specific process collaborators. The user can select the process collaborator to investigate with a drop-down menu. The rightmost card shows the absolute uncertainty (ALU) in the lane, the relative lane uncertainty (RLU), and the lane uncertainty balance (LUB). The second card shows a bar chart. This chart illustrates the distribution of uncertainties grouped by trust concerns for the selected collaborator. The third card illustrates dependency-related metrics, such as data influence, data dependency, message influence, and message dependency in text boxes. The last diagram depicts the distribution of process elements for the selected collaborator.

In the upper right corner of this page, the user can select a perspective. The perspective can assume any trust persona. By selecting the trust persona to investigate, Trust Studio reduces the global trust metrics according to the trust tolerance profile of the persona. The pdf-button lets the user export a trust report. The trust report is a machine-generated description of the trust metrics. At the bottom of the page, the user can see a list of all uncertainties and their characteristics.

Configuration In Trust Studio, the user can define the input parameters of TAPE as illustrated in Figure 4.5.

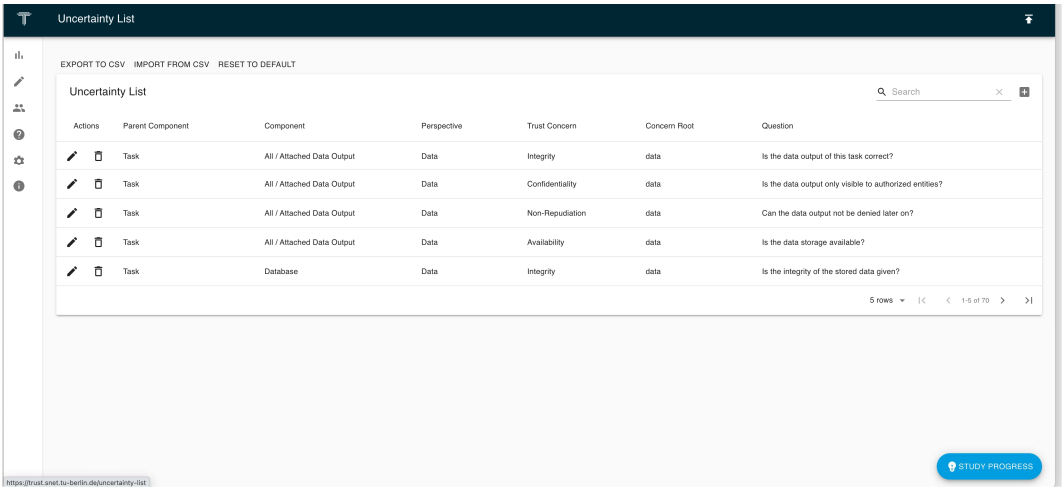

Figure 4.5: Trust Studio user interface: TAPE configuration.

An editable table lets the user modify the uncertainty possibility list. The process components, uncertainty root, and trust concern can be selected with a drop-down.

Apart from these static parameters, Trust Studio also enables the user to set and modify the trust tolerance profiles of certain trust personas. Figure 4.6 shows how the user can define new trust policies. The trust tolerance profile of a trust persona consists of a set of trust policies. The user can add or delete trust policies in a table. The data depicts the trust entity, process element, and trust concern defining the trust policy.

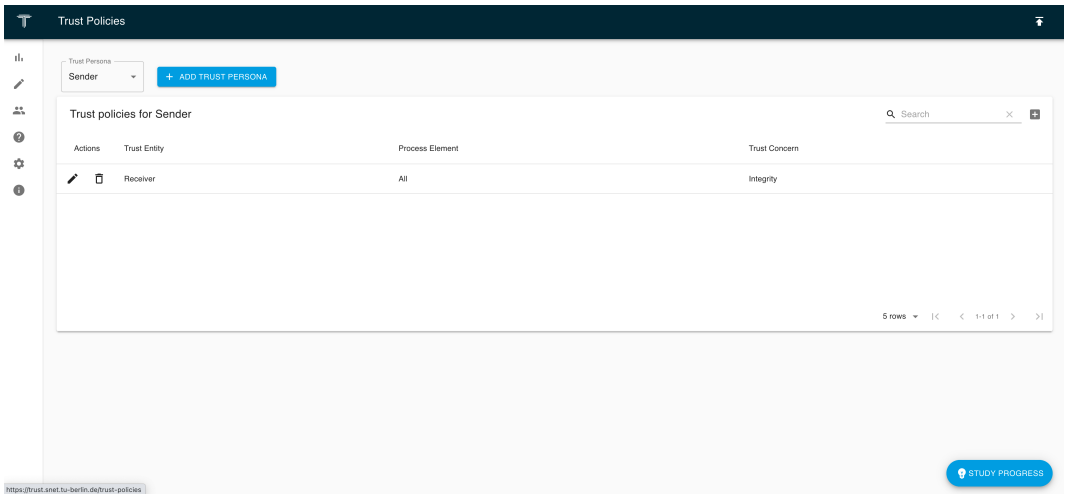

Figure 4.6: Trust Studio user interface: trust policies.

4.5 Summary

This chapter introduced Trust Studio as a tool to support the application of the TAPE method.

- Trust Studio can be used in two different use cases. Section 4.1 discussed the descriptive and the creative use casee in detail.
- Section 4.2 elicited functional and non-functional requirements for the implementation of Trust Studio.
- Section 4.3 discussed the architecture of Trust Studio as a software system.
- Finally, Section 4.4 discussed the implementation of the Trust Studio software tool.

The outcome of this chapter and the concepts in Chapter 3 established the primary design science artifacts that this thesis proposes. The next chapter evaluates these artifacts regarding their utility.

5 Evaluation

This chapter evaluates the solution approaches from Chapter 3 and 4. These aim to be *minimal viable artifacts* to analyze and mitigate trust issues. Thereby, the evaluation assesses their utility using different methods. Furthermore, this evaluation aims to identify weaknesses in the introduced concepts. These insights pose the starting point for future work.

Generally, the scientific contributions of this work fall into the broad category of *design science* [19] in the context of information systems research. The design science research paradigm aims to improve the capabilities of humans or organizations to reach their goals by creating innovative new artifacts. From a methodological viewpoint, the design science research paradigm is fundamentally a *problem-solving* paradigm [187]. In information systems research, design science aims to solve a problem with *IT artifacts*. These IT artifacts may be "constructs (vocabulary and symbols), models (abstractions and representations), methods (algorithms and practices), and instantiations (implemented and prototype systems)" [19]. Concerning the scientific contributions of this work, the solution approaches can be classified as follows:

- **Method**: TAPE is a new method to analyze and mitigate trust issues in collaborative business processes automatically using business processes.
- **Instantiation (software tool)**: Trust Studio is a supporting artifact for the application of TAPE to concrete use cases. The automated tool makes it easier for process engineers and analysts to apply the TAPE method.
- **Constructs**: The introduced trust patterns are constructs that show how trust issues can be mitigated with distributed ledger technologies.

Gregor and Hevner [188] propose to classify knowledge contribution of design science artifacts according to their application domain maturity and solution maturity, as illustrated in Figure 5.1. This work's contributions can be interpreted as an improvement to the current state of the art of trust in collaborative business processes. Trust issues in collaborative business processes are a well-known problem across different research fields. The introduced artifacts for analyzing and mitigating trust issues are novel approaches. Furthermore, many scientific publications incorporated distributed ledgers to improve some trust issues in processes. Yet, the classification and formalization of different distributed ledger trust patterns were not formalized before this work.

The goal of the design science research paradigm is to provide *utility* through the construction of artifacts. Hence, the overall objective of evaluating the scientific contributions of this work is centered around utility. Figure 5.2 shows the high level evaluation outline. The figure shows that the solution approaches

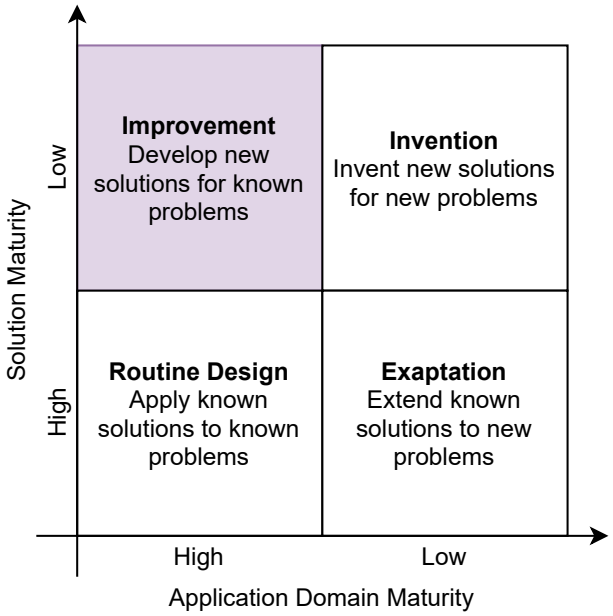

Figure 5.1: Classification of contributions according to the design science research knowledge contribution framework by Gregor and Hevner [188].

lead to predictions (or claims) regarding their utility. The evaluation methods themselves include *ex ante* and *ex post* approaches. The evaluation is predominantly qualitative. In general, the evaluation follows the principles of evaluating artifacts in design science as proposed by Johannesson and Perjons [189].

Regarding the TAPE method, the prediction is that TAPE helps process engineers and analysts to understand and improve their trust issues in their processes better. Therefore, two distinct evaluation methods aim to validate this prediction. A conceptual-methodological analysis in Section 5.1 analyzes TAPE's metrics. In addition, the section discusses conceptual limitations of the method in a discussion. A controlled experiment with a group of experts analyzes the ease of use and usefulness of TAPE and Trust Studio in Section 5.2.

The prediction related to Trust Studio is that the software artifact is a sufficiently performant and usable tool that aids in applying the TAPE method. Trust Studio is tightly coupled with the TAPE method that it implements. Thus, the controlled experiment in Section 5.2 provides insights regarding both, the TAPE method and Trust Studio as an implementation thereof. In addition, a performance evaluation of Trust Studio conducts a dynamic analysis of the execution of Trust Mining based on a sample set in Section 5.3.

The distributed ledger trust pattern classifications led to the prediction that distributed ledgers can mitigate trust issues in real-life use cases. Therefore, a collection of different case studies in Section 5.4 illustrates their applicability and utility.

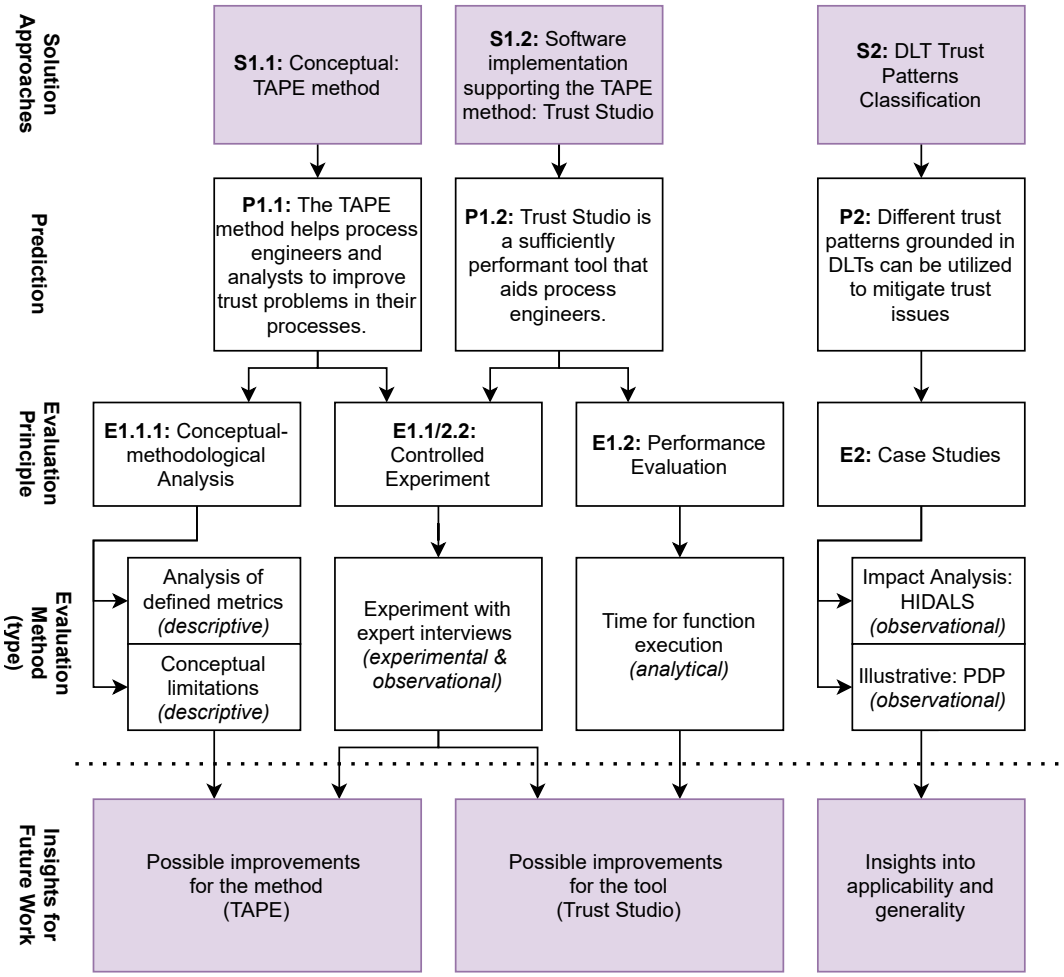

Figure 5.2: Evaluation methodology of this thesis.

5.1 Conceptual-methodological Analysis of the TAPE Method

This section provides a conceptual discussion of the TAPE method. Therefore, the first paragraphs discuss conceptual limitations inherent to TAPE. The analysis of the defined trust-related metrics in the later paragraphs demonstrates TAPE as an artifact on a set of sample business process models. This demonstration illustrates how the metrics translate to real-world processes and how they represent their characteristics.

5.1.1 Conceptual Limitations of the TAPE Method

Flexibility and Extensibility TAPE is designed in a way that it can be configured, modified, and extended to enable different usage patterns. TAPE's underlying trust model can be configured concerning its inherent semantics. Users may change trust concerns and uncertainty roots according to their focus of observation. Thus, it is possible to utilize TAPE for different purposes. For instance, a user might not want to observe a particular trust concern. In this case, the process engineer may delete the uncertainty possibilities regarding

the particular trust concern from the uncertainty possibility list before applying Trust Mining. The same principle may be applied to extend the UPL. The provided reference set of uncertainties is non-exhaustive and designed to be adapted to different situations.

As a second foundation, TAPE builds upon a process model. The past chapters utilized the MCBPM meta-model and BPMN as illustrations. Yet, in general, it is possible to apply the discussed concepts to any process modeling language that addresses the basic modeling capabilities of activities, swimlanes, data, and process flows.

It is possible to extend TAPE semantically with a *context property*. A trust context enables more fine-granular modeling and descriptions of process elements. Context properties may express that an "activity uses standard software" or "source code is open-source". With this context property, trust policies can be defined that enable a richer semantic analysis. For instance, a trust persona might have a trust policy that standard software or open-source code is always trusted. The context property may be of relevance in any of TAPE's three major steps. The specification and extension of TAPE with a context property are subject to future work.

Furthermore, also future work may extend the individual three main steps of TAPE. For instance, this may happen by adding new steps to the core workflow. It is also possible to modify existing steps. For example, in the third step of Trust Mining, future work can add new metrics to analyze other trust-related aspects.

Abstraction Bias The analytic parts of TAPE face major challenges regarding varying levels of abstraction within the subprocesses of different organizations. In inter-organizational collaborations, cooperating organizations typically have limited knowledge and insights into the processes of other organizations. For example, in cross-company supply chains utilized in the concept in Chapter 3, the activity to deliver the parcel is modeled as one single task. The single task represents an *outside perspective* onto the processes. In reality, this activity may be a larger subprocess. Within a logistics company, different employees may pack, load, and track the parcel. For competitive reasons, they may hide their internal activities to the external collaborators.

Different levels of abstraction threaten the usefulness of the uncertainty annotation. For instance, the "deliver parcel" activity is modeled as a collapsed subprocess hidden to all other collaborators. In this case, Trust Mining annotates at maximum every uncertainty possibility once to that subprocess. This leads to an absolute lane uncertainty equal to the maximum number of uncertainty possibilities defined for an activity. Alternatively, process engineers can model the process so that three activities replace the subprocess. This modeling decision leads to each of these activities getting annotated with separate uncertainty possibilities. This leads to an ALU of three times the maximum number of uncertainty possibilities for an activity. In such a situation, also other metrics, such as the RLU, are artificially increased. Compared to the other collaborators, this might indicate a particular high uncertainty balance towards the organization

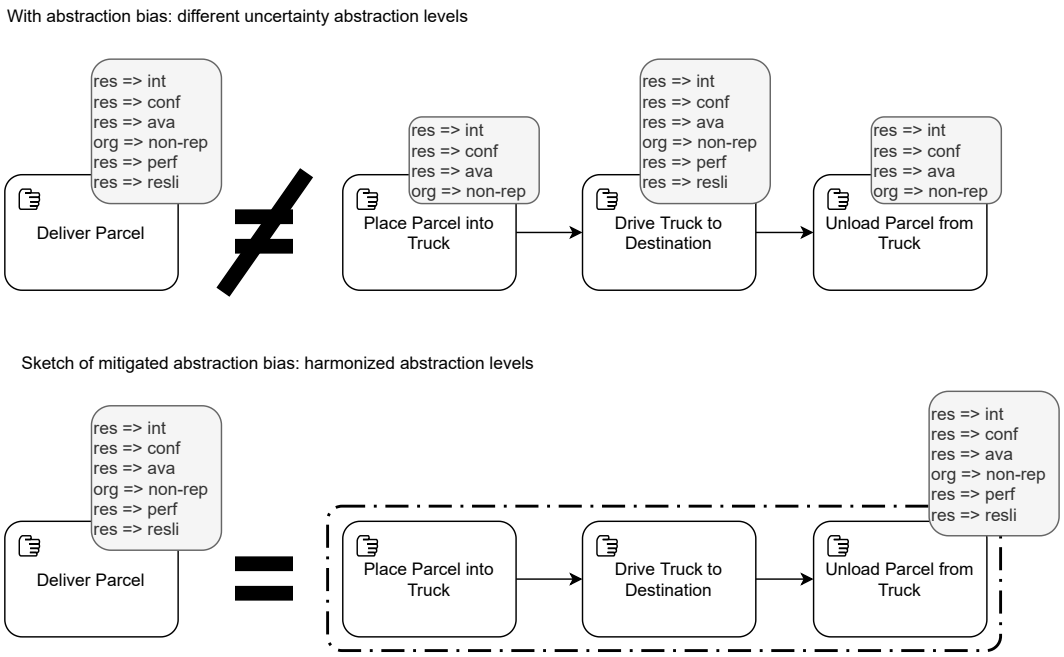

Figure 5.3: Illustration of the abstraction bias regarding uncertainty annotation an a possible solution sketch.

with more fine-grain modeling of uncertainties. The following sections refer to this phenomenon as the *abstraction bias*.

The abstraction bias may distort the comparative metrics in Trust Mining. In its minimum viable artifact version, the method cannot automatically address the abstraction bias. The most obvious way to mitigate the abstraction bias is to ensure that the process as a whole is modeled on the same level of abstraction. Yet, this approach leads to a lot of manual work that is error-prone.

Another possibility to mitigate the abstraction bias is to group activities to indicate that they are one "uncertainty unit". Therefore, future work may extend the underlying process model with a way to group activities. For instance, the BPMN language has the possibility to group process elements. Future work may employ the grouping syntax to represent uncertainty units. Figure 5.3 sketches the principle. In general, also other approaches to mitigate the abstraction bias are imaginable. Thus, future work needs to create and compare different approaches. There may be different optimal solutions depending on the situation. Without either future work to mitigate the abstraction bias conceptually or ensuring the same abstraction level manually, the abstraction bias poses a threat to the validity of the concepts related to Trust Mining and TAPE.

Systematic Limitations Conceptually, TAPE builds upon a trust model and a process model. Even though the utilized models can be changed, TAPE requires an underlying process model. Hence, TAPE cannot analyze and mitigate trust-related aspects of a process that cannot be represented in its process model. This circumstance delineates a systematic limitation of TAPE.

Trust Mining introduces relationship metrics. They are based on the interaction of different process components of separate organizations. Yet, Trust

Mining cannot analyze trust relationships established implicitly outside the process. For instance, Trust Mining enables its users to analyze a situation where one organization depends on data coming from another organization. Yet, the concept can hardly analyze a situation in which a trust persona trusts a process collaborator implicitly due to their relationship. For example, one organization might trust another organization and its subcontractors. In TAPE, process engineers can model this situation by introducing *explicit* trust policies. However, it is not possible to automatically derive them from the *implicit* relationship of these organizations. The following paragraphs refer to this phenomenon as *relayed trust*. Future work may create concepts to analyze this apart from introducing trust policies.

Trust Mining traverses the process model *piece-wise* to annotate uncertainties. This annotation limits the focus of observation to only one process element at a time. If two activities are intended to be performed in parallel, the current version of Trust Mining only analyzes each of them separately. Yet, there might be uncertainty across components, such as if both activities will conclude before a given deadline. This is called a *cross-component uncertainty*. Future work needs to focus on an extension of the initial Trust Mining concept to analyze these dependencies semantically.

The presented trust analysis concepts do not differentiate between different "trust weights". Thus, uncertainties have equal weight. This design choice places trust-aware BPM as an alternative to risk-aware process BPM. Risk-aware BPM always assigns risk probabilities and an impact quantification to activities. The product of these two real numbers are a metric for the overall risk. Thus, for risk-aware BPM, it is necessary that the process engineer can *assess* these risk values confidently. In collaborative processes, separate organizations may not have insights into the detailed subprocesses of the others. Thus, risk assessment may not be possible. The trust-aware approach is an alternative to the risk-centered perspective. However, process engineers and analysts may still desire to have very *rough* notions of different levels of trust. A trust-aware view on business processes does not need to assess that an uncertainty caused by one organization has a probability of 23.2%. Yet, a *relative* notion that one trust persona trusts one organization generally "more" than another organization might be beneficial in a trust-centered approach. These "trust levels" are subject to future work.

The application of the introduced trust patterns also has limitations. The discussed taxonomy provides process engineers with an approach to describe and classify their trust-enhancing capabilities. Their selection might be more complex in real-world scenarios. Typically, trust patterns modify the process. Thus, newly introduced elements may also create new uncertainties. The choice of trust patterns often depends on external factors. These considerations may include the cost of implementing and executing the trust pattern, its complexity, or technical limitations in the adaptability of the current system. Thus, future work may establish a more complex decision model that selects the optimal trust patterns for trust issues.

Syntax and Semantics TAPE is a mostly syntactical approach to analyze and mitigate trust issues in a process. The semantics of the trust relationships and uncertainties are injected through the UPL in Trust Mining and the process engineer's interpretation. For instance, an uncertainty in a process regarding the integrity trust concern within an activity can be described with the question "is the activity executed correctly?". In the example of a "deliver parcel activity", this can be translated as "is the parcel delivered correctly?". When creating an invoice after a job has finished "is the invoice created correctly by the system" would be the semantical equivalent. Both semantics can be described with the most general question "is the activity executed correctly?". However, the translation to the context of a certain process component may still be semantically non-trivial.

TAPE aims to be semantically as general as possible to be applicable to a wide variety of potential business processes. At the same time, it is specific enough so that the process engineer can easily derive value from the trust analysis. In the presented version, the translation from the generic trust semantics to the concrete context of a process poses an entry barrier for applying TAPE. The way of thinking to understand the trust issues semantically may be non-trivial for many process engineers and analysts. Future work may add easier-to-understand semantics to TAPE. Educating the process engineers and analysts to apply the required mental framework may also lower the entry barrier.

5.1.2 Analysis of TAPE Metrics

The TAPE method introduces metrics that aim to quantify trust-related concepts. The metrics aim to enable process engineers to better understand trust on a theoretical level in a process. The following paragraphs apply Trust Mining and the metrics to a set of 137 BPMN models. Aggregated observations into the distribution of these metrics allow to analyze their utility.

Data Set The collection of 137 BPMN collaboration diagrams originates from three different open-source repositories [190, 191, 192]. The initial set of diagrams was filtered to obtain a subset that is compatible with the meta-model. Therefore, only syntactically valid BPMN diagrams are utilized for this demonstration. They must have at least two different organizations since collaboration with oneself is semantically not a collaboration. The BPMN diagrams need to be compatible with the MCBPM meta-model. Thus, organizations need to be modeled as pools. Pools are the BPMN equivalent of the inter-organizational swimlanes in MCBPM.

Furthermore, performing semantical analysis leads to a set of BPMN diagrams that can be considered "meaningful". Within the scope of this work, meaningful BPMN diagrams have no disconnected elements as well as start and end events for all process flows. Furthermore, meaningful diagrams must also not perform implicit gateways. Finally, meaningful BPMN diagrams need to adhere to a set of linting rules as a best practice for meaningful BPMN diagrams[1]. Applying

[1]see https://github.com/bpmn-io/bpmnlint/tree/master/docs/rules for more details

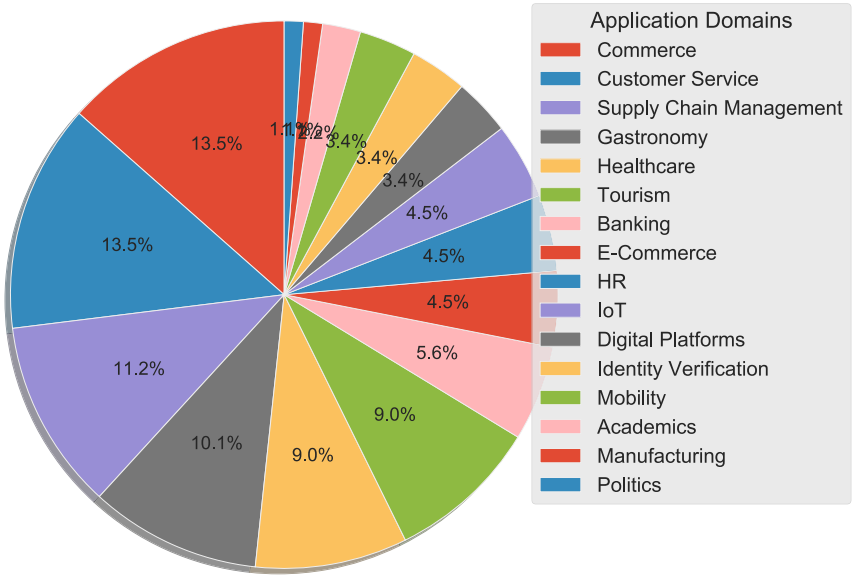

Figure 5.4: Application domains of the 137 test BPMN diagrams for the demonstration of trust-related metrics.

these requirements to the initial set of BPMN diagrams leads to the set of 137 collaboration diagrams that are subject to the analysis of the TAPE metrics.

The reference data set consists of process models from 16 different application domains. Figure 5.4 shows the distribution of BPMN models and their application domains. They include (but are not limited to) e-commerce, finance, HR, and healthcare processes.

Prior to quantitatively analyzing the TAPE metrics, this section describes the features of the processes according to the general metrics proposed in [193]. Figure 5.5 shows the distribution of certain feature occurrences across all process models. As illustrated in the upper left corner, the diagrams have an average of 11 activities per process model. The mean total number of gateways (TNG) is 2.74 (min. 0, max. 38), while the mean total number of events (TNE) is 1.05 (min. 0, max. 17). The generalized events describe different types of intermediate events. TAPE is only helpful to analyze inter-organizational processes. Thus, the preprocessing of the data sets was carried out, under the assumption that everybody trusts themselves. Hence, all considered diagrams have at least two organizations. This leads to a mean number of pools is 3.03 (min. 2, max. 14). The MCBPM meta-model does not distinguish between inter-organizational swimlanes (BPMN calls them pools) and intra-organizational departments and units (BPMN calls them intra-organizational lanes). The data set also shows that the pools are often used without further splits into intra-organizational lanes (number of lanes NL, min. 0, mean 1.56, max. 7). The upper right corner shows that the diagrams have an average of 0.95 data objects per process (min. 0, max. 13). Hence, this data set employs modeling data objects in processes sparsely. The two remaining metrics show the connectivity level of activities (CLA, min. 0.3, max. 1, mean 0.59) and the connectivity level between pools (CLP, min. 0, max. 4.5, mean 1.41).

The argument of why this data set is suited for demonstrating the TAPE metrics unfolds as follows. This work introduces TAPE and distributed ledger

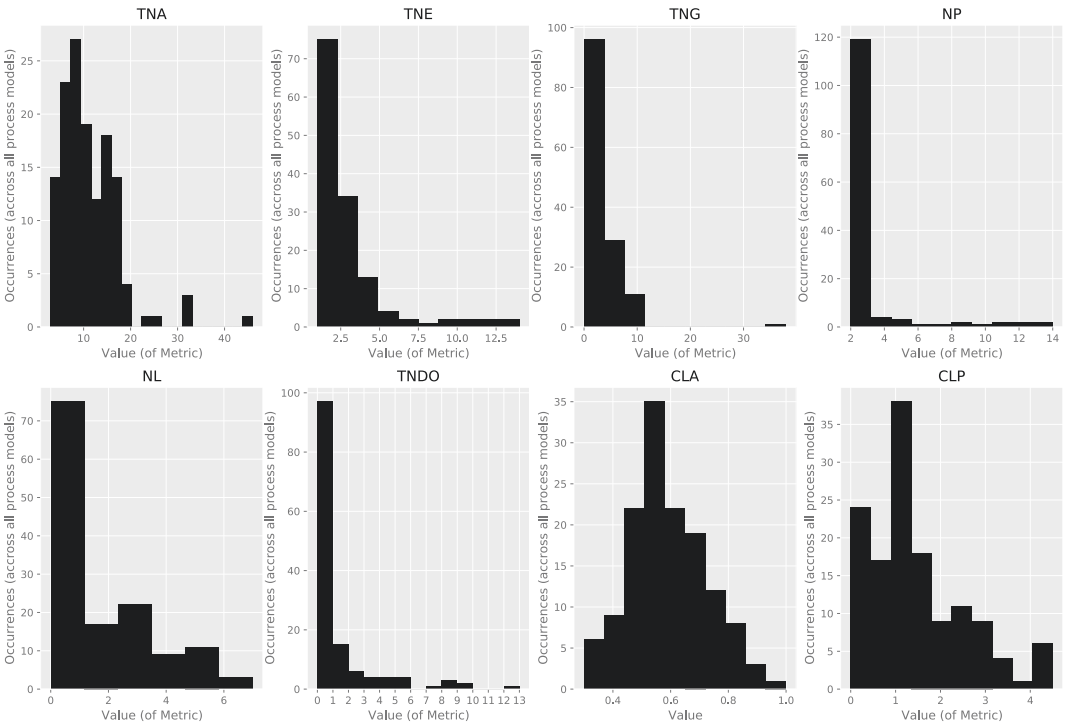

Figure 5.5: Static characteristics of 137 BPMN diagrams utilized for the demonstration of TAPE metrics. The x-axis shows the value of the metric. The y-axis shows how often the value occurred for the metric across all process models. TNA: total number of activities, TNE: total number of event, TNG: total number of gateways, NP: number of pools, NL: number of lanes, TNDO: total number of data objects, CLA: connectivity level between activities, CLP: connectivity level between pools.

trust patterns as minimal viable artifacts. Thus, 137 diagrams as the data set for the minimum viable artifacts can be considered sufficiently large for the first evaluation. Furthermore, the features show that BPMN models are compatible with the MCBPM meta-model. Nevertheless, most of the data sets do not utilize data objects. Thus, it is only possible to validate the metrics related to the data flow in Trust Mining to a limited extent.

TAPE Metrics Figure 5.6 illustrates the introduced TAPE metrics in an example setup without any trust policies. Trust policies add a specific perspective on trust in a process. This demonstration of the TAPE metrics aims for quantitative analysis. Since all of the utilized process models have different organizations, they are not comparable to each other. Hence, the following discussion focuses on the global perspective of the trust metrics concerning their explainability and utility.

As seen in the upper left corner, the mean global uncertainty is 146 (min: 13, max: 862). Attributing the uncertainties to different swimlanes leads to a mean average lane uncertainty (ALU) of 55 (min: 4.67, max: 172.4). The ALU has a correlation with the number of pools. The majority of the 137 sample diagrams include two or three pools. Dividing the mean global uncertainty by the mean number of pools results in 55 average uncertainties per lane. The upper right diagram shows the mean relative lane uncertainty (RLU) distribution. It is

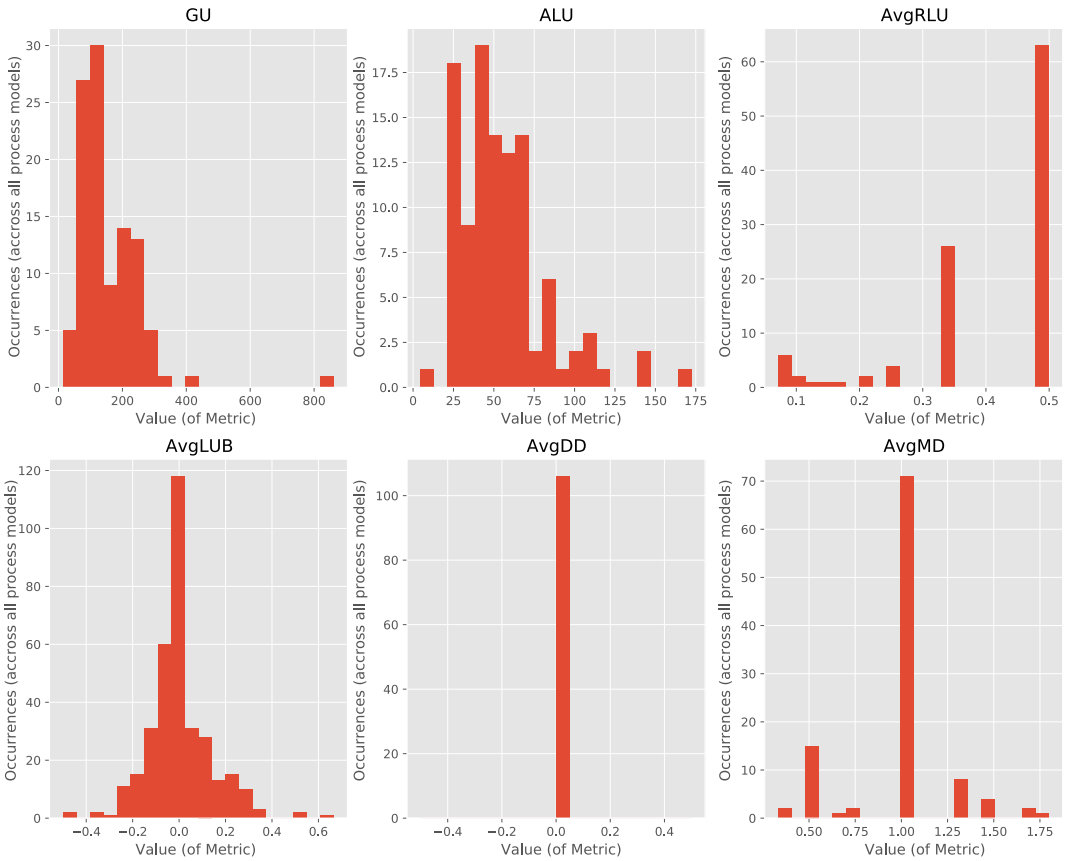

Figure 5.6: TAPE characteristics of 137 BPMN diagrams utilized for evaluating Trust Mining. The x-axis shows the value of the metric. The y-axis shows how often the value occurred for the metric across all process models. GU: Global uncertainty, ALU: average lane uncertainty, AvgRLU: average relative lane uncertainty, AvgLUB: average lane uncertainty balance, AvgDD: average data dependency, AvgMD: average message dependency.

visible that most of the processes have an RLU value that is either close to 0.5 or close to 0.33. The histogram of the average lane uncertainty (LUB) balance shows a peak at 0. The calculation of this metric was done by enumerating all uncertainty balances of all lanes in all processes. This peak indicates that the data set has characteristics where the uncertainty between the different lanes is mostly balanced. The spread shows a minimal value of -0.5 and a maximum value of +0.6.

The bottom center histogram shows that the majority of the diagrams in the sample set have an average data dependency of 0. The 137 test process model rarely use data objects. Process models with no data objects always have a data dependency of 0. This infrequent use of data objects in the evaluation data set is also visible in the TNDO histogram. Thus, interpreting the values of data dependency is only possible to a limited extent. This limits the generality of this evaluation. The lower right figure shows the average message dependency (MD). In this histogram, a mean of 1 for message dependency is visible. This is explainable by the fact that the diagrams often use the request-response message pattern [37]. In this pattern, messages to a single recipient are responded with a single answer, hence balancing the message dependency.

Results The observation of TAPE metrics on a sample set of business process models enables assessing the utility of these metrics. Therefore, widespread distribution of different metric values can be interpreted as an indicator for a good metric due to its high discriminatory characteristics. A metric that would map many inputs to the same value shows a lower discriminatory capability.

The metrics of GU, ALU, and AvgLUB calculated on the 137 process models show a widespread distribution of different metric values. Also, the metrics for AvgRLU and AvgMD show a wide variety of different metric values with a few dominant clusters. Thus, these metrics are arguably suited to represent distinctive trust-related characteristics. The AvgDD metric shows the same value for the majority of the test process models. This can be explained by the rare use of data objects in the input models. Thus, at this point, it is not possible to judge the utility of the data dependency metric.

5.2 Controlled Experiment for TAPE

This section describes a controlled experiment to validate how the TAPE method helps process engineers improve trust issues in business processes. Therefore, the focus of the controlled experiment is to assess the *perceived usefulness* of the TAPE method and the *perceived ease of use* of TAPE and Trust Studio.

5.2.1 Experiment Setup

The controlled experiment consists of three major phases, illustrated in Figure 5.7. Each experiment involves three actors. The *guiding researcher* explains the principles of the experiment and the design science artifacts. The *experiment participant* experiences the introduced methods and tools in a hands-on fashion. The *observing researcher* collects data during the experiment execution.

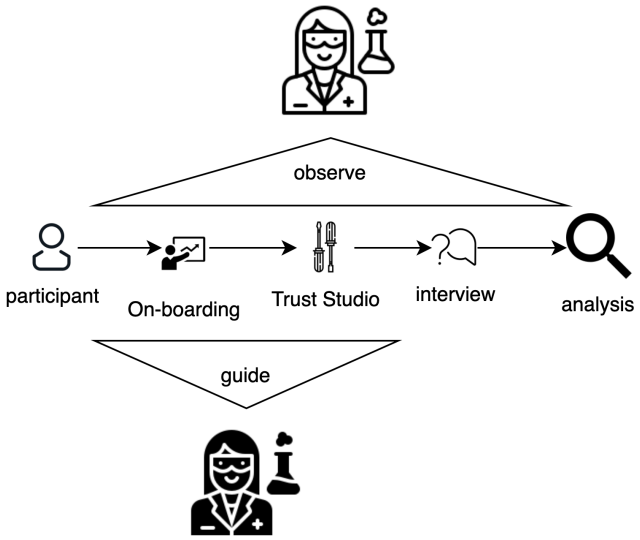

Figure 5.7: Sketch of the flow of the controlled experiments.

At the beginning of each experiment, the test subject receives a short on-boarding session. This session communicates the principles of TAPE, Trust

Studio, and trust patterns. After onboarding, the participant enters the execution phase of the controlled experiment. This phase comprises two different parts. In the analytical part, the guiding researcher introduces the participant to the Trust Studio application. Therefore, the participant uses a sample process model that the researcher provides. In the constructive part, the experiment participant gets in touch with a set of trust patterns classified according to the TAPE method. This part of the experiment provides the test participant with a list of trust patterns. The participant's task is to find trust patterns suited for mitigating trust issues in the given process. After the execution of the experiment, a final interview takes place. This interview reflects on the experiences of the participant. It aims to assess the perceived usefulness and ease of use of the introduced concepts. In addition, the feedback from the test participants serves as the basis to identify possible extensions, improvements, and future work on TAPE.

The experiments are executed in recorded online sessions using the video conferencing tool Zoom[2]. The interaction with the Trust Studio is facilitated utilizing an online version of the tool. In that way, participants have minimal setup effort. The maximum duration for the controlled experiment is 90 minutes per participant.

5.2.2 Experiment Flow

The following sections discuss in detail the experiment flow and outcomes. Figure 5.8 serves as reference for the *experiment script*. The full script for the experiment is documented in Annex B. For the experiment, ten test participants were interviewed. The majority of the selected participants have some basic understanding of business processes and distributed ledgers. Yet, no participant was an expert in business process management. The set of test participants was equally divided between industry practitioners and research-oriented test subjects. The industry practitioners are software, system, or industrial engineers. In their daily work live, they have to deal with collaborative processes to some extent. The research participants were all active in computer science subfields, including distributed systems, information systems engineering, and privacy. Except for one industry practitioner, all test participants had a general understanding of blockchain and distributed ledger technologies.

5.2.3 Onboarding

The onboarding session starts with a short outline of the experiment and its four phases as illustrated in Figure 5.8. Afterward, the experiment participants are asked about their previous experience in formal models for business processes. Depending on their experience, a short explanation about BPMN is provided.

Afterward, the participant is introduced to the example business process shown in Figure 5.9[3]. This process is the reference that the participant uses throughout the experiment's analytic and constructive execution phases.

[2] https://zoom.us

[3] The processes is modeled with the BPMN modeling language. For the remainder of this thesis, the event-based gateway can be triggered multiple times. This convention is done for clarity of the illustrated process and is an extension to the strict definition of the BPMN 2.0 standard.

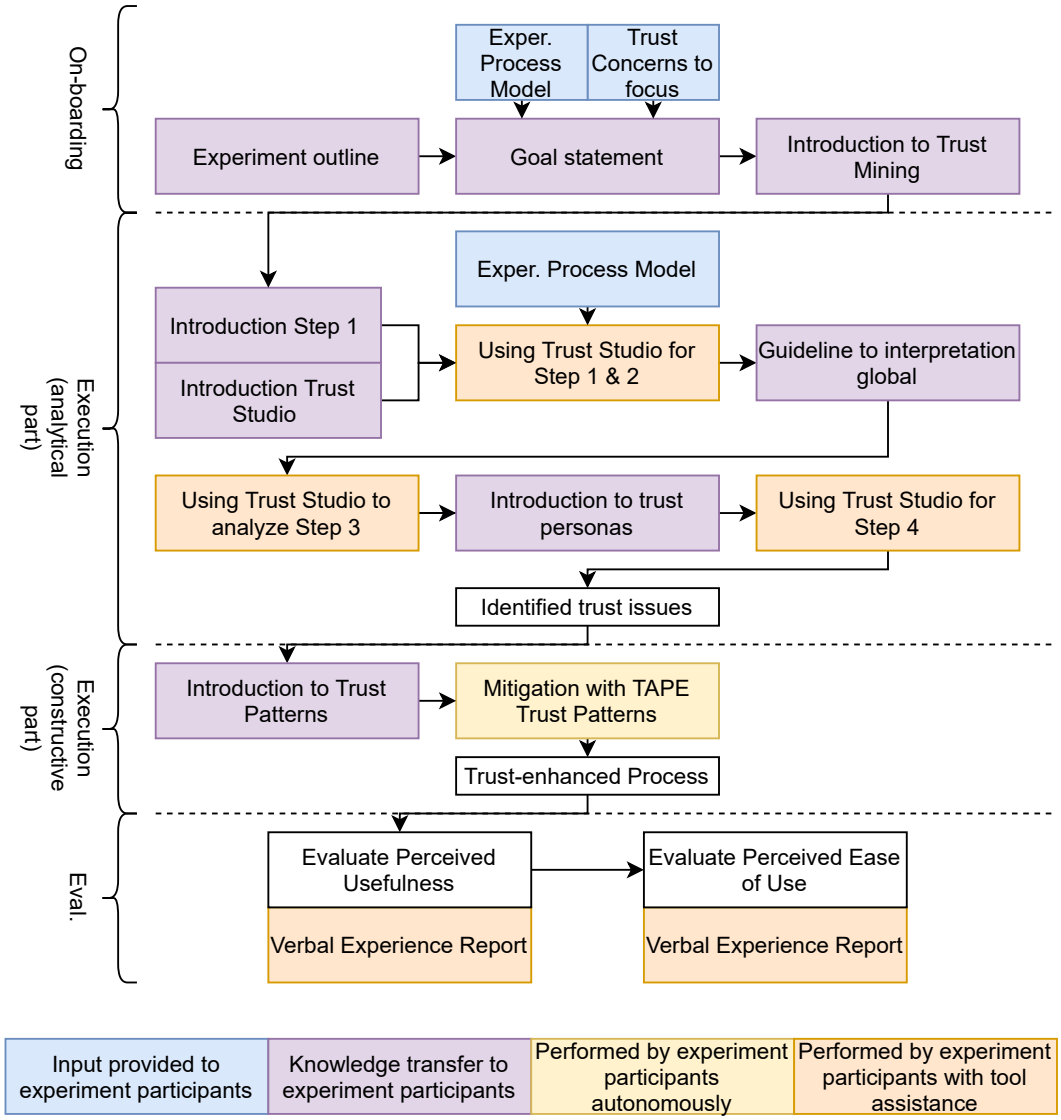

Figure 5.8: Detailed experiment flow of the controlled experiment.

The reference business process depicts a peer-to-peer crowdfunding workflow. The test participant is asked to imagine that a friend forwarded the link to a crowdfunding campaign. The hypothetical campaign creator is an engineer who built a prototype of wireless in-ear headphones with an AI. The AI is capable of translating languages in real-time. In that way, the user can receive translations of a foreign language in real-time through the headphones, similar to the Babel Fish in *Hitchhiker's Guide to the Galaxy* [194].

The process includes two different organizations. The *campaign creator* wants to collect funds for the large-scale production of the first batch of Babel Fish headphones. A *backer* is a person looking for smart ways to invest money and receive useful gadgets. In general, there can be many backers. The business process model focuses on one specific backer. A collapsed pool illustrates that an arbitrary number of backers can invest in the campaign. Yet, the process flow is the same for all backers. The process starts with the campaign creator setting a funding goal and a campaign end. In the Babel Fish example, the campaign creator seeks to raise 30,000 Euro until July 4th. Therefore, the creator

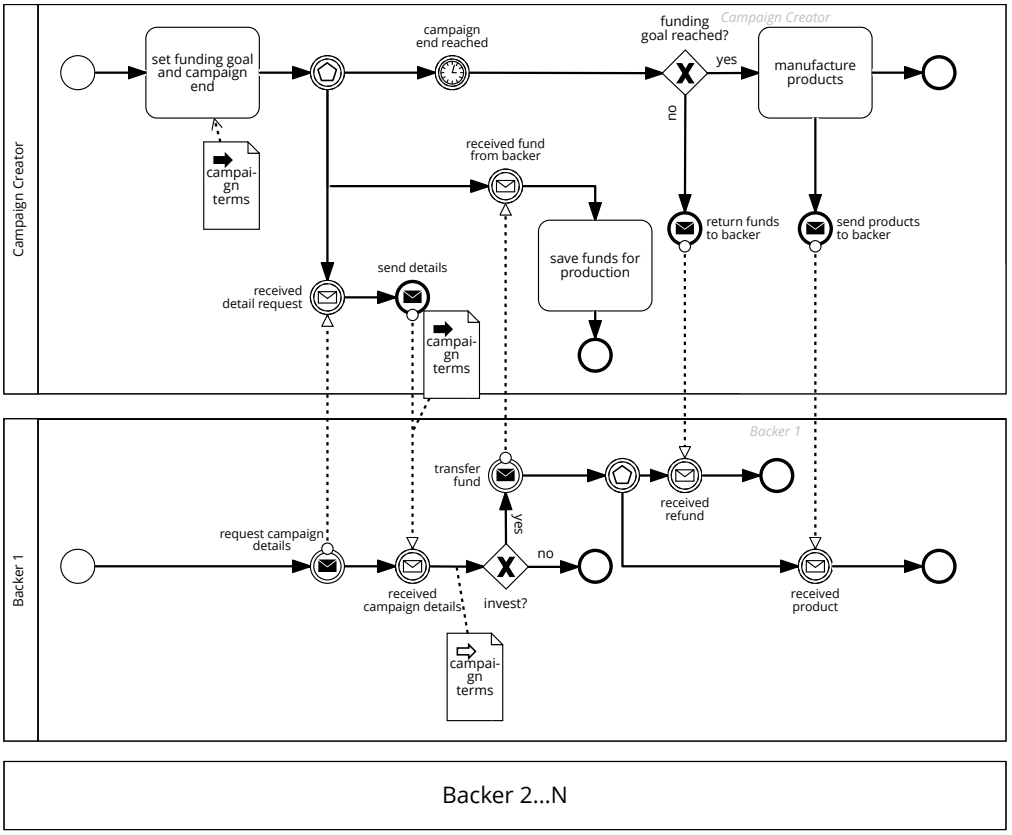

Figure 5.9: Process for controlled experiment: peer-to-peer crowdfunding.

aims to collect 300 Euro from each backer to finance the batch production in a Chinese factory. The campaign creator states these conditions in the campaign terms. The terms also ensure that backers will be reimbursed with their entire investment if the funding goal is not reached by the end of the deadline. When a backer considers investing in the Babel Fish headphones, the backer requests these funding terms from the campaign creator. The campaign creator sends the campaign terms document to the backer. The backer analyzes the document and decides whether to participate. In case the backer chooses not to participate, the process ends. Otherwise, the backer transfers 300 Euro to the campaign creator's checking account. The campaign creator transfers the money to a dedicated subaccount until the deadline passed. If the backers did not meet the funding goal before the deadline ended, the campaign creator returns all funds to all backers. If the campaign creator achieved the goal, the campaign creator makes the order to manufacture the products. Afterward, the campaign creator distributes the first batch to the backers. For a 300 Euro investment, every backer is eligible to receive one set of Babel Fish headphones.

After introducing the business process, the test participant receives a short overview of the concept of trust concerns. The guiding researcher briefly explains integrity, confidentiality, availability, non-repudiation, performance, and resilience with examples. The experiment participant has the opportunity to ask questions regarding the concepts to the executing researcher. With these two

inputs, the test subject is confronted with the goal statement of identifying and mitigating trust issues in the presented process. A short introduction to Trust Mining follows the goal statement.

5.2.4 Execution (Analytical Part)

In the analytical part of the execution phase, test participants use TAPE and Trust Studio to analyze trust issues in the given process. The guiding researcher provides the experiment participant with a short explanation of Trust Mining, its use, and how it is automated within Trust Studio.

The analytical parts starts with an introduction to Trust Studio. The interactive tutorial introduces the user to the user interface elements and explains where the participant can find different aspects of TAPE. Throughout the tutorial, the experiment participants learn the fundamental interactions with Trust Studio. These include uploading process models, creating trust personas, or using the dashboard with the introduced trust metrics. During the tutorial, the test participant has the opportunity to ask additional questions.

After introducing the Trust Studio tool, the experiment participant is tasked to analyze the trust issues of the given crowdfunding example. Therefore, the executing researcher provides the participant with an XML BPMN file of the process. Thus, there is no need to model the process from scratch, eliminating a potential error source. The experiment participant is asked to navigate to the modeler section of Trust Studio, where the annotated process model is shown. The guiding researcher asks the participant to analyze and interpret the uncertainty annotation of the "save funds for production" activity and the "funding goal achieved" gateway. Therefore, the participant is asked to iterate over the annotated trust concerns and translate them into semantic questions. For instance, a human resource causing uncertainty regarding the integrity trust concern in the "funding goal achieved" gateway would translate to "is the decision to either reimburse backers or take the funds for production done correctly?". At this point in the experiment, the test participant has a short discussion with the executing researcher, explaining their interpretation. The observing researcher takes notes to assess the interpretations of the participant.

The next step in the analytical part of the experiment focuses on the trust metrics in the dashboard. Therefore, the guiding researcher provides a guideline to the global trust metrics. The test participant is tasked to examine the dashboard and interpret the metrics for the crowdfunding process. During this phase, the test participant reports findings and questions regarding the concepts that could not be comprehended.

As the last task in the analytical part of the experiment, the test participant has to model a new trust persona in the process. The chosen trust persona for the exerpiment is the backer. After a short introduction, the test participant creates the new trust persona with two example trust policies. Then, the test subject is asked to go back to the dashboard and select the backer's perspective from the trust persona dropdown. A new graph appears dividing between "critical" (i.e., trust problems for which a trust policy was introduced) and "uncritical"

uncertainties. Finally, the guiding researcher and the test participant discuss the participant's interpretation of the trust personas and their policies.

5.2.5 Execution (Constructive Part)

In the constructive part of the controlled experiment, the participant applies one exemplary distributed ledger trust pattern to mitigate a trust issue. All participants should focus on the "save funds for production" activity.

This phase starts with an initial introduction to the concept of trust patterns. Therefore, the guiding researcher provides the test participant with a table of trust patterns. The table contains the distributed ledger trust patterns discussed in Chapter 3. The columns of this table show the different dimensions of classification. These include:

- the pattern name,
- the trust-enhancing method,
- the process element,
- the trust concern,
- the uncertainty root,
- a short description of the pattern, and
- an illustration of the pattern with a process model.

The experiment participant is asked to express the uncertainty that the campaign creator might use the backer's funds to buy a sports car and vanish. Therefore, the participants classify this uncertainty with its trust concern, the root of uncertainty, and the process element. Then, the test subject is asked to iterate over the provided trust pattern table and identify potential trust patterns that can mitigate the trust issue. For the technical details of the trust pattern, the test participant can ask the guiding researcher questions.

The constructive part of the controlled experiment focuses on a single trust pattern and trust issue to get mitigated. This restriction was introduced to enable the execution of the entire workflow within 90 minutes.

5.2.6 Interview

The interviews at the end of each controlled experiment are the primary means of data collection for the evaluation. The test subject has to answer a set of questions to reflect their experiences. Methodologically, this approach is based on qualitative version the Technology Acceptance Model (TAM) [195]. The TAM measures the acceptance of a technology based on its *perceived usefulness* and *perceived ease of use*. It focuses on the acceptance of technology rather than on the acceptance of a method. TAM is based on the Theory of Reasoned Action (TRA). This approach is a general model "designed to explain virtually any human behavior" [196].

Thus, there are similarities between the acceptance of a model (such as TAPE) and a technology. Perceived usefulness can be divided into three clusters of questions for its assessment: job effectiveness, productivity, and importance of the system to the execution of the job. These three clusters are arguably also of relevance for a method. Also, perceived ease of use can be divided into three parts: physical effort, mental effort, and ease of learning. These clusters can be argued to be of relevance for a method. Therefore, the application of the TAM to the TAPE method in the controlled experiment is argued to be viable.

Based on these principles, the TAM questions are adapted to the outline of assessing a method. Table 5.1 shows the 12 questions used in the interviews of the controlled experiments. The setup for the interview questions was adapted to cover the situation that the test participants have only used TAPE and Trust Studio in one experiment. This setup stands in contrast to the traditional TAM. There, the questionnaire is usually conducted after the test subject has used the new technology for a longer time in the day-to-day job.

5.2.7 Results

This section describes the results of the controlled experiment. The analysis uses two primary data sources. As a first data source, the analysis discusses the participant's reactions throughout the experiment. The answers to the interview questions, as shown in Table 5.1, are the second major data source. The insights derived from this analysis pose the starting point for potential improvements and future work.

Onboarding In the onboarding section of the controlled experiment, the guiding researcher introduced the test participant to the process model. Afterward, the test participant had the opportunity to ask clarifying questions regarding the process. Seven out of ten test participants had no further questions. Two test participants asked specifics about the used BPMN notation. These test subjects had not used BPMN before and asked about the semantics of the event-based gateway and the triggering of events. One test participant needed clarification that the example process is executed entirely in a peer-to-peer fashion with no centralized platform as an intermediary. After introducing the process model, the guiding researcher introduced the test participants to the six trust concerns. All test participants were familiar with the concepts and had no further questions.

Analytical Part In the analytical part of the experiment execution, the participants got in touch with an online version of the Trust Studio software. The initial tutorial included visual and textual descriptions and explanations by the guiding researchers. This tutorial also introduced the concepts of uncertainties and the four steps of Trust Mining. Two participants asked questions after the tutorial. One participant asked where she could find different "perspectives" in the process. This test subject did not understand the concept of trust personas

	Question	Focus	TAM-equivalent statement
1	Is trust an aspect that you currently cover in your job? If yes, do you think trust studio would help you to accomplish your tasks more quickly?	PU	Using Trust Studio for trust analysis would enable me to accomplish my tasks more quickly
2	How crucial do you think trust is for typical business processes?	PU	Using Trust Studio would improve my job performance.
3	Which tools or methods to analyze trust do you use currently and do you think Trust Studio would increase your productivity?	PU	Using Trust Studio in my job would increase my productivity
4	Do you think using Trust Studio would enhance your effectiveness?	PU	Using Trust Studio would enhance my effectiveness on the job.
5	Do you think using Trust Studio would make it easier for you to identify trust-related issues?	PU	Using Trust Studio would make it easier to do my job.
6	Do you find Trust Studio useful?	PU	I would find Trust Studio useful in my job.
7	How did you like the onboarding experience?	PEoU	Learning to operate Trust Studio would be easy for me.
8	Is Trust Studio's navigation easy? Is it easy to gain insights into the trust issues in Trust Studio?	PEoU	I would find it easy to get Trust Studio to do what I want to do.
9	How complex do you think are the interactions?	PEoU	My interaction with Trust Studio would be clear and understandable.
10	What interaction use cases can you envision with it? Analytical, constructive?	PEoU	I would find Trust Studio flexible to interact with.
11	What do you think have you not „mastered" yet?	PEoU	It would be easy for me to become skillful at using Trust Studio.
12	Is Trust Studio easy to use?	PEoU	I would find Trust Studio easy to use.

Table 5.1: The table illustrates the questions used in the interviews, their focus (PEoU: perceived ease of use, PU: perceived usefulness), and the equivalent question in a strict TAM questionnaire.

and trust policies directly and needed more clarification. One participant clicked on the wrong button at first, and the tutorial did not start.

After the initial tutorial, the participants were asked to upload a provided BPMN diagram in XML format using the upload button. Nine out of ten subjects faced no issues with this task. One participant tried to drag and drop the file to Trust Studio. This functionality is not yet implemented. Thus, the test participant had to reload the page. In the next step, the participants needed to click on the pen icon for analyzing the uncertainty annotation. While observing the uncertainty annotation, the guiding researcher discussed the trust concerns' semantics. For this purpose, the researcher used the "save funds for production" activity and the "funding goal reached" gateway as a reference. Half of the participants had no other questions, whereas the other half needed short discussions on the semantics of the uncertainties. Interpreting the meaning of the uncertainty triplets was initially hard to understand for these participants.

Next, the guiding researcher tasked the test participants to go to the dashboard and observe the introduced metrics. Seven of them navigated to the dashboard without any issues. Two participants asked reassuringly that the right button is selected, and one could not find the navigation to the dashboard without the help of the guiding researcher.

The following step in the experiment had the test subject examining the dashboard's graphs and metrics. Two participants stated that the colors of the uncertainty distribution chart are hard to distinguish. Four test participants appeared overwhelmed with the dashboard and the collection of metrics. They needed some clarifications on the semantics of the metrics. The most common question aimed to understand the triplet of the root, trust concern, and process element that characterizes an uncertainty.

After observing the dashboard, the guiding researcher asked the test participants about their trust policies. All test participants responded with situations where they would not trust somebody. This is the opposite of how TAPE defines trust policies. In TAPE, trust policies express a *positive* relationship, i.e., who trusts whom for what. The test participants stated *negative* relationships, i.e., what their trust issues are. Four test participants stated trust issues that relied on *context* that were not modeled in the process diagram. For instance, one subject stated that her trust policies are different when 300 Euro are at stake compared to 30 Euro. Another common remark was about the context of the execution of certain activities. For example, one participant stated that she would trust the software the campaign creator uses. Yet, that participant would not trust the employee to use the software correctly. Since the process model included a single activity for this logic, this cannot be distinguished.

The next step in the experiment included modeling trust policies for the process. Therefore, the participants used the forms to add new trust policies. Eight of the ten participants initially pressed the wrong button. Instead of creating new trust policies, they attempted to create new trust personas. The button for creating new trust personas was visually highlighted. The button for adding a trust policy, on the other hand, was smaller and marked in a muted color. This lead to visual confusion for the participants. The participants were tasked to

model a trust policy where the backer trusts the correct triggering of events by the campaign creator. Semantically, this was explained with the situation that most events are message events. The backer assumes the communication for the messages is trustworthy (via e-mails or comparable tools). One participant had a typographic error in the word "campaign", which led to Trust Studio not recognizing the trust subject correctly.

After modeling the trust policies, the guiding researcher asked the participants to go back to the dashboard and observe the influence of the defined trust policies on different trust personas. Five out of ten test subjects switched back and forth between the trust personas using the dropdown menu. Three participants did not directly understand the influence of trust policies on the relevance of specific trust issues. After short explanations, all participants understood the introduced principles. Observing the trust personas' relevant trust issues marked the end of the analytical part of the experiment.

Constructive Part In the constructive part of the experiments, the test subjects were asked to state their most critical trust issues. The experiment participants identified the top trust issues as getting money back when the funding goal was not achieved or stealing the money without receiving anything. The guiding researcher explained that the participant has to observe the "save funds for production" activity more closely in the next step. Given that the campaign creator might execute the activity incorrectly, the test subjects were asked to translate this uncertainty into the correct triple of uncertainty root, trust concern, and process element. The participants were advised to use the uncertainty-annotated process model in Trust Studio to do this task. Initially, two participants selected the non-repudiation trust concern instead of the integrity trust concern. Five test participants needed reassurance regarding the semantics of the three different dimensions, stating they forgot their meaning. In the end, all participants translated the trust issue correctly to the three dimensions.

After this conceptualization, the participants received a table with a classification of different trust patterns. The researcher asked them to observe the columns for activity, the root of uncertainty, and trust concern. The goal was stated to match the trust pattern with the previously identified trust issue. All participants selected the smart contract activity correctly, regardless of their background knowledge in distributed ledgers. This also holds for the one test subject that has never got in touch with DLTs and blockchain. Afterward, a short discussion regarding the meaning of the smart contract activities followed.

Finally, the participants were asked if they think their trust issues are now better. All of them answered that they think so. Two participants stated that the trust patterns introduce new trust concerns. For example, one test participant asked if she can trust the DLT. Another test participant stated that also the implementation of the trust pattern is a trust pattern itself. With that statement, the constructive part of the experiment concluded.

Interview The final interview aimed to reflect on the test participants' perceived usefulness and ease of use. The first six questions are derived from the

TAM reference questions regarding the perceived usefulness of the experienced methods.

1. *Is trust an aspect that you currently cover in your job? If yes, do you think Trust Studio would help you to accomplish your tasks more quickly?* All test participants stated that trust is essential in the processes they are involved in within their professional life. Furthermore, all participants stated that Trust Studio would speed up their analysis process of trust.

2. *How crucial do you think trust is for typical business processes?* Similar to the first question, all participants stated that they think trust is crucial in business processes with different collaborating organizations.

3. *Which tools or methods to analyze trust do you use currently?* Eight of the ten test participants answered that they do not use any tool to analyze trust in their process. One participant stated to use automatic fraud detection tools to analyze the trustworthiness of payment information in her job. This industry practitioner is working in e-commerce. Another test participant expatiated upon company policies and contractual agreements for collaboration with external partners in her organization. She identified the willingness to sign such agreements as a measurement of the trustworthiness of the business partner.

4. *Do you think using Trust Studio would enhance your effectiveness?* All test participants stated that using Trust Studio would enhance their effectiveness. Seven participants argued that the tool helps with the completeness of trust issues. They stated to do the entire process of analyzing potential trust issues in their head is difficult and error-prone. Trust Studio enables completeness since it is automated.

5. *Do you think using Trust Studio would make it easier for you to identify trust-related issues?* The answers regarding this interview question were diverse. One participant argued that she would have achieved the same results using a purely ad-hoc method. Two participants stated that they had to receive more training to identify the meaning of the trust-related issues more. In addition, two other participants stated that Trust Studio helps mostly with localizing trust issues. Interpreting them, on the other hand, is a rather challenging task.

6. *Do you find Trust Studio useful?* The direct question regarding the perceived usefulness of the tool was answered positively by all participants. Two test subjects expressed their desire to receive more training.

The second six questions are derived from the TAM reference questions regarding the perceived ease of use of the experienced methods.

7. *How helpful was the onboarding experience?* The experiment participants generally regarded the onboarding experience as appropriate and helpful. One participant reported that the onboarding steps included too much text and expressed the desire for a faster process.

8. *Is Trust Studio's navigation easy?* Regarding Trust Studio's navigation, eight participants reported an easy-to-understand workflow. Three participants expressed their confusion about the trust policies table and stated that there could be some improvements regarding the clarity of the buttons. Tooltips were not part of Trust Studio. Two participants noted that the addition of tooltips would be beneficial. One test subject who worked with SAP tools for long periods of her carreer stated that the tool is complex yet still more straightforward to learn than most SAP software.

9. *How complex do you think are the interactions in Trust Studio?* No experiment participant reported overly complex interactions in Trust Studio. One participant stated that it is easy to make mistakes with the trust policies. More pre-filled dropdowns would be helpful.

10. *What use cases can you envision with it?* Most participants stated that the analytical and constructive use cases should always be executed consecutively. Three participants also introduced a new "explain" use case. The explain use case is helpful in meeting-like situations where different stakeholders discuss the trustworthiness of a process. Those three test participants stated that they are frequently involved in such discussions and the Trust Studio would be helpful.

11. *What do you think have you not "mastered" yet?* The answers to that question were diverse. One reported a need for more training on the relationship graphs. Three participants stated that they are not familiar with the process modeling language and require more training to use the tool properly. Two test subjects stated that they need some practice in their daily tasks with the tool. One test participant said that she requires more training on the uncertainty possibility list and how to modify it for different focuses of the trust analysis task.

12. *Is Trust Studio easy to use?* Six experiment participants answered without any conditions that Trust Studio is easy to use. Three stated that they think they need more training. One test participant reported that Trust Studio is not intuitive.

5.2.8 Insights and Threats to Validity

The controlled experiment is a qualitative evaluation of the perceived usefulness and ease of use of TAPE and Trust Studio. The experiment included a limited sample set of ten test participants. Thus, the controlled experiment cannot provide comprehensive insights into the generality of TAPE. Yet, the experiments could identify some action points for future work.

The experiment showed that understanding the concepts behind TAPE is essential for the use of Trust Studio. Future work should focus on building new approaches to an onboarding experience. Improvements regarding the user interfaces and interactions would aid the general usability. Concluding, the experiments showed that all test participants derived value from the application of TAPE using Trust Studio. Even the test participant, without any knowledge about

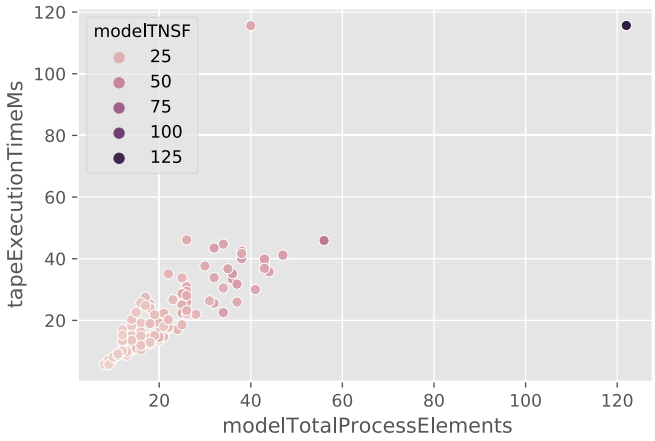

Figure 5.10: TAPE trust analysis execution time scatter plot. The x-axis depicts the number of process elements of the input model, the y-axis shows the execution time. The hue illustrates the number of sequence and message flows in the model.

DLTs, was able to identify the correct trust pattern to mitigate the trust issue. This may lead to the interpretation that the presented approach adds a layer of abstraction to a process that helps non-technical users find approaches to trust issues they could not find without the method. Thus, this thesis presented a *minimal viable method* that offers benefits for its users. The generality of these insights needs to be validated in future quantitative studies.

5.3 Performance Evaluation of Trust Studio

Practically useful design science artifacts need to offer acceptable performance. Thus, this section describes a performance evaluation of Trust Studio as the implementation of TAPE. This section analyzes the execution time of Trust Mining as the primary performance metric. The execution time of the automated trust analysis is the only part in TAPE that does not involve live interaction by a user.

The previously discussed prototypical implementation of Trust Studio implements all algorithms of Trust Mining, metrics, and trust persona models in a web application using JavaScript. The following execution time analysis refers to the total time of executing all steps of Trust Mining without utilizing any trust policies.

Figure 5.10 shows the execution time of annotating the sample data set of 137 process models. This includes computing all TAPE metrics for every process model. The performance evaluation was executed on a machine with a 2.3 GHz Quad-Core Intel Core i5 processor and 8 GB RAM. In general, the execution time increases linearly with an increasing number of process elements and sequence flows between them. This reflects the design of Trust Mining. The approach traverses the process model iteratively several times. For the uncertainty annotation, the process model needs to be traversed once and check at maximum every uncertainty possibility for its relevance at the current element. The UPL is a list of constant length. Thus, this leads to an algorithmic complexity of $\mathcal{O}(n)$,

	tapeExecutionTimeMs
count	106.0
mean	22.94
std	16.67
min	5.67
25%	13.18
50%	18.41
75%	28.53
max	115.65

Table 5.2: Statistics on TAPE metrics of the 137 input BPMN diagrams.

where n is the sum of process elements. Regarding the relationship analysis, the data dependency analysis has to compare each data object with every other data object in conjunction with the UPL. Since the UPL has constant length, this leads to a worst-case complexity of $\mathcal{O}(n^2)$.

Observing the execution time of Trust Mining on the 137 BPMN diagrams in Figure 5.10 shows that the execution time increases with the number of process elements. The increase follows a linear trend, even though there is some variance visible. This trend is explainable due to the rarely utilized data objects sample set. Therefore, the computation of the trust metrics does, in most cases, not involve a quadratic component.

Overall, the performance evaluation shows an execution time between 5 and 40 milliseconds for most of the models. Thus, the execution of Trust Mining is nearly instant. Hence, it is fast enough to be a useful tool for trust analysis that does not require the user to wait for long-lasting and expensive computations.

5.4 Case Studies for DLTs as Trust Improvement

This thesis introduced distributed ledger trust patterns as new constructs to mitigate trust issues in collaborative business processes. The following sections demonstrate the design science artifact "in action" by applying the TAPE method to two real-world case studies. HIDALS is the first case study from the supply chain domain, similar to examples already shown in Chapter 3. PDP is the second case study originating from the medical domain. Insights derived from these case studies are used to reason about the generality of TAPE, the impact of DLTs as trust patterns, and future work.

5.4.1 HIDALS

In today's global value chains, different logistics organizations have to collaborate to deliver goods from a sender to a receiver. For instance, the German national carrier, Deutsche Post, does not offer services in the Netherlands. Senders often define Service Level Agreements (SLAs) for high-value packages. For instance, high-value pharmaceuticals need to stay cold throughout their delivery.

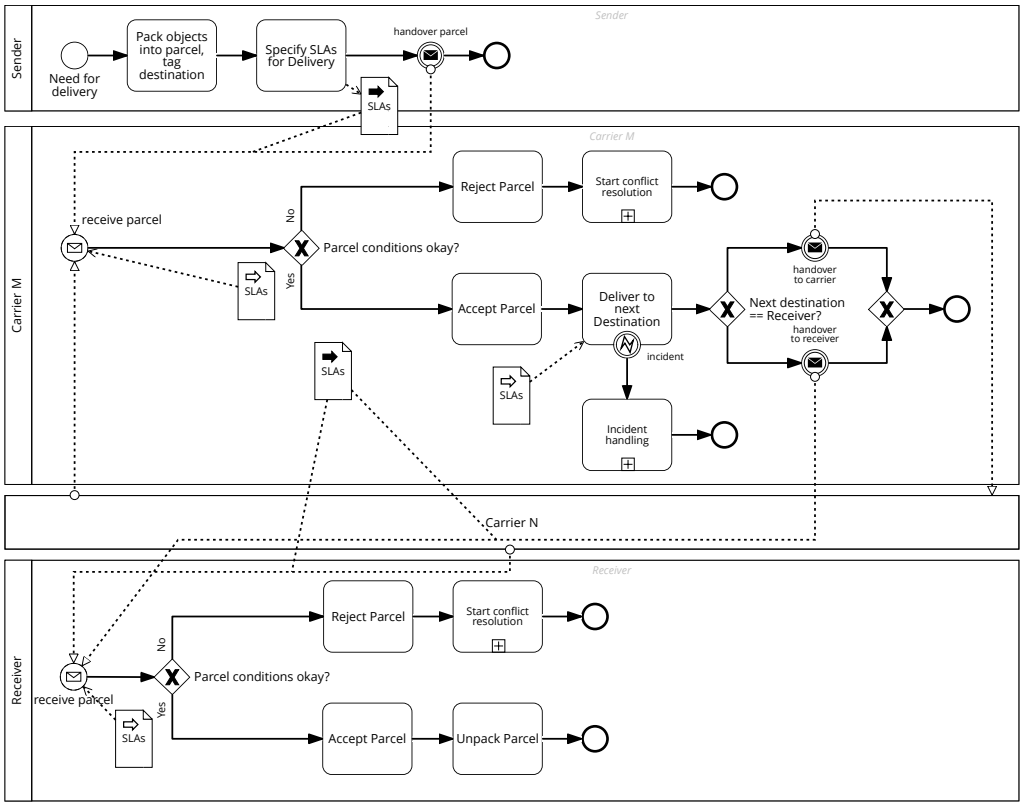

Figure 5.11: Initial As-Is process of the Gravity supply chain project.

All involved logistics organizations need to adhere to the SLAs in an end-to-end process. However, validating the adherence to SLAs in inter-organizational processes is challenging. Today's end-to-end delivery services lack transparent and trustful mechanisms to keep track of deliveries beyond organizational limits.

The HIDALS system (Hybrid IoT-based Decentralized Application for Logistics Supply Chain Management)[4] aims to address these problems by using distributed ledger technologies and autonomous sensors in a cross-company collaboration. The following section discuss the HIDALS system in a case study. The case study has been carried out within the innovation activity "Gravity" funded by the European Institute for Innovation and Technology (EIT) in 2019 and 2020.

As-Is Process and Trust Issues The Gravity project aims to increase the trustworthiness of international supply chains of high-value, fragile, or perishable goods. Use cases for international deliveries with such goods include pharmaceutical supply chains (e.g., COVID-19 vaccinations), delivery of livestock (e.g., bumblebees for fertilization), or fragile technical equipment (e.g., microscopes).

The delivery process for these use cases is comparable to the running example in Chapter 3 and is depicted in Figure 5.11. The process starts with a sender who

[4]Results for HIDALS have been pre-published in the following two conference papers: Müller, M., Garzon, S. R., Westerkamp, M., & Lux, Z. A. (2019, October). HIDALS: A Hybrid IoT-based Decentralized Application for Logistics and Supply Chain Management. In 2019 IEEE 10th Annual Information Technology, Electronics and Mobile Communication Conference (IEMCON) (pp. 0802-0808). IEEE. and Müller, M., & Garzon, S. R. (2019, August). Blockchain-Based Trusted Cross-organizational Deliveries of Sensor-Equipped Parcels. In European Conference on Parallel Processing (pp. 191-202). Springer, Cham.

needs to send a high-value dose of cancer medicine to a customer. The drug is a particular make. Only a few units are available worldwide. The sender's factory is located in Rome, Italy. The receiver is an oncologist in Berlin, Germany, who ordered the dose for her patient. The patient has a unique disease pattern. The oncologist can only treat this pattern with the rare, high-value medication from the pharmaceutical supplier in Italy. The delivery process starts with the sender packing the medicine into the parcel, tagging its destination, and specifying SLAs. The cancer medicine is in a small box, which must not be dropped. In addition, the medication needs to be kept cold. Its temperature must never exceed 24 degrees Celsius. Specifying these SLAs in the As-Is process involves filling out a document. The sender hands this document over with the parcel to the carrier. This handover starts the delivery.

In contrast to the running example in Chapter 3, the delivery chain consists of many (n) carriers that are involved in the delivery. An international supply chain like the cancer medicine example includes several legs. The first leg is the transport by an Italian carrier (e.g., Poste Italiane). This first carrier delivers the parcel to Rome-Fiumicino airport. Upon arrival, the Italian carrier hands over the parcel to an airfreight carrier (e.g., DHL). The airfreight carrier transports the parcel on a flight to Berlin-Brandenburg International airport. Arriving there, the airfreight carrier hands over the parcel to the last-mile carrier (e.g., Deutsche Post). The last carrier delivers the parcel to the receiver.

Formally, the process consists of legs connected by handovers. In the process model in Figure 5.11, message exchanges indicate handovers. The SLA data object is attached to the message exchange. This represents the communication of SLAs between the two organizations. A carrier receiving a parcel analyzes superficially if the parcel seems to be in an acceptable condition without opening it. In case the receiver of a handover rejects a parcel, an external conflict resolution process is started. Conflict resolution is not modeled in the diagram. In case a carrier accepts the parcel, the carrier delivers it to the next destination. If the carrier detects an incident, for example, dropping the parcel and hearing the fragile contents shattering, the carrier must open an incident in the carrier's system. Handling that incident, e.g., by paying out compensation for the damage, is another external subprocess. Once the carrier arrived at the destination, another handover is made. In case this is an intermediate handover, another carrier is the handover partner. The handover process is similar for all carriers. Thus, the BPMN diagram only illustrates carrier m in a full pool, while all other n carriers are modeled with a collapsed pool.

In case the last carrier reached the receiver, the handover is different. The receiver takes up the parcel, validates its conditions by opening it, and analyzes its content. If the parcel is not acceptable, the receiver rejects it, and another conflict resolution subprocess starts. If the parcel is accepted, the receiver unpacks it, and the delivery process terminates. In the pharmaceutical example, the doctor distributes the medicine at this point to the cancer patient.

In the requirements phase of the Gravity project, different stakeholders identified three major problems. The first is the assurance of *adherence to service level agreements*. In such an inter-organizational process, the delivery subprocesses

	Initial **General**	Initial **Receiver TP**		Initial **General**	Initial **Receiver TP**
Global Metrics					
GU	191.00	191.00			
AEU	3.60	3.60			
UU	0.00	36.00			
RU	191.00	155.00			
Reg. Sender			**Reg. Carrier N**		
LU	36.00	0.00	LU	0.00	0.00
RLU	0.19	0.00	RLU	0.00	0.00
LUB	−0.06	−0.33	LUB	−0.25	−0.33
DI	2.00	2.00	DI	0.00	0.00
DD	0.00	0.00	DD	0.00	0.00
MI	1.00	1.00	MI	2.00	2.00
MD	0.00	0.00	MD	1.00	1.00
Reg. Carrier M			**Reg. Receiver**		
LU	100.00	100.00	LU	55.00	n.a.
RLU	0.52	1.00	RLU	0.29	n.a.
LUB	0.27	0.67	LUB	0.04	n.a.
DI	0.00	0.00	DI	0.00	n.a.
DD	1.00	1.00	DD	1.00	n.a.
MI	2.00	2.00	MI	0.00	n.a.
MD	2.00	2.00	MD	2.00	n.a.

Table 5.3: The tables shows the trust metrics from the general perspective and from the perspective of the receiver trust persona (receiver TP) regarding the global metrics, the sender, carrier m, carrier n, and the receiver.

GU: global uncertainty, AEU: average element uncertainty, UU: uncritical uncertainty, RU: relevant uncertainty, LU: total lane uncertainty, RLU: relative lane uncertainty, LUB: lane uncertainty balance, DI: data influence, DD: data dependency, MI: message influence, MD: message dependency.

of the single carriers are independent of each other and cannot be influenced. As a second major issue in the process, the stakeholders identified the *verification of SLA violations*. For instance, detecting that a parcel with perishable goods got too warm is a challenge in the current process. The last major problem is *assigning responsibility to SLA violations*. When a receiver realizes that the cancer medicine is unusable because it got too warm, large economic damage may be caused. Even though the receiver is insured via the primary carrier, this primary carrier aims to get reimbursed by the carrier in the supply chain that caused the violation initially. These three problems were unsolved issues for the process stakeholders at the beginning of the Gravity project.

Applying TAPE Step 1: Identifying Trust Issues This paragraph illustrates the application of Trust Mining in TAPE's first step to the supply chain process. Applying Trust Mining yields the metrics shown in Table 5.3. From a general perspective (without any trust personas), there are 191 uncertainties in the process which distribute to an average of 3.6 uncertainties per process element. Distributing the uncertainties among the collaborators shows that the sender is responsible for 19% (RLU) of the uncertainties, carrier m for 52%, carrier n for 0%, and the receiver for 20%. Carrier n is modelled as a collapsed pool and can be neglected. The distribution of uncertainties shows a significant imbalance towards carrier n (+0.27).

No.	uncertainty root	trust concern	process element	description
1	resource	integrity	activity	handling the parcel so that no SLAs are violated
2	resource	integrity	event	emitting the incident event when a SLA violation occurs
3	organization	non-rep.	event	not being able to deny a SLA violation at the end of the process

Table 5.4: Trust issues selected for mitigation in the Gravity project.

The receiver trust persona has a good relationship with the sender. The doctor from Berlin and the pharmaceutical specialist from Rome have been working together for a long time. Thus, from the perspective of the receiver trust persona, the sender is trusted for all uncertainties. Formulating this constraint as trust policies shows an even larger imbalance in the process. From the 191 initial uncertainties, only 155 are relevant for the receiver trust persona since she trusts the sender fully and trusts itself. That leaves with all the relevant uncertainties in the process being attributed to the carriers.

Trust Mining leads to a list of uncertainties that can be mitigated. Previously, the process stakeholders identified three major issues in the process that can be found and classified with the uncertainty annotation. The first issue is the adherence to SLAs. This can be classified as a resource (uncertainty root) causing uncertainty regarding the integrity (trust concern) of the "deliver to next destination" activity (process component). The second trust issue is verifying that SLA violations are identified and compensation tasks are started correctly. TAPE classifies this as the resource (uncertainty root) causing uncertainty regarding the integrity (trust concern) of the incident error event (process element). The final trust issue is assigning the responsibility of SLA violations to different carriers. TAPE formalizes this situation as the organization (uncertainty root) causing uncertainty regarding the non-repudiation (trust concern) of the incident error event (process element). These three uncertainties are the most critical ones for the process stakeholders. They are subject to uncertainty mitigation in TAPE's second step.

Applying TAPE Step 2: Mitigating Trust Issues with DLT Trust Patterns Based on the three classified trust issues, the second step of TAPE selects trust patterns to mitigate the issues. In the Gravity project, the introduction of the HIDALS system mitigates the trust issues. HIDALS uses three major trust patterns:

1. **Automation**: IoT devices monitor the conditions of the parcel on the way. Whenever the IoT devices detect an SLA violation the event gets triggered.
2. **Decentralization**: An independent entity owns the IoT devices. This party is solely responsible for managing and maintaining IoT devices. In that way, reporting incidents is decoupled from the organization that potentially causes the incidents and would have an incentive to deny it.

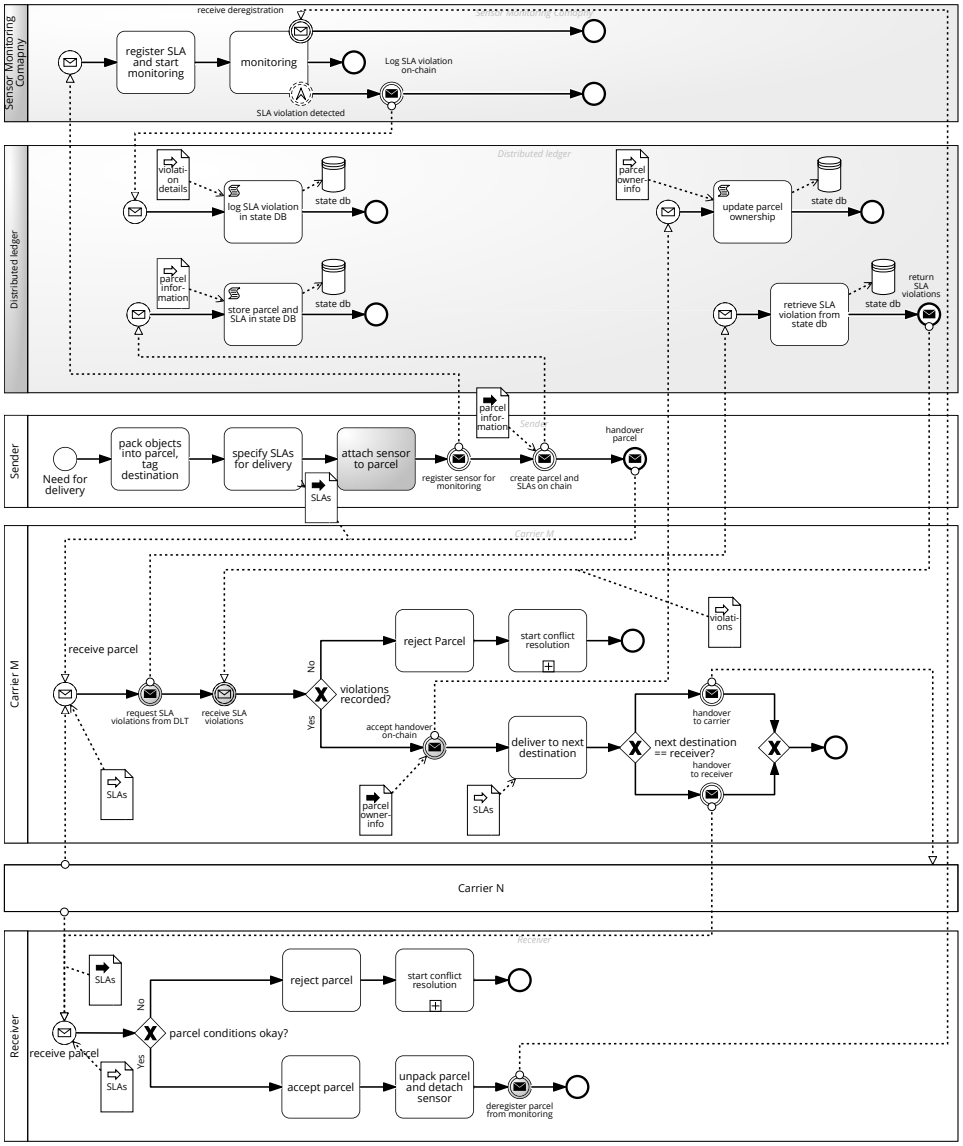

Figure 5.12: Trust-improved process of the Gravity supply chain project with the HIDALS system. Changes compared to the As-Is process are marked in grey.

3. **Transparent Event Log**: The HIDALS system introduces a distributed ledger to store the possession of parcels and SLAs related to it. Whenever an event gets triggered, it is stored in a transparent event log on-chain, so non-repudiation is assured.

Applying these trust patterns leads to the trust-enhanced process model in Figure 5.12. The process starts again with the sender tagging parcel and specifying SLA. In contrast to the initial process, the next step is attaching a physical IoT sensor to the inside of the parcel. Within the Gravity project, small *cubes* with temperature, acceleration, humidity, and light sensors were used, see Figure 5.13. The cubes include modems that enable wireless connection via the Narrowband-IoT protocol.

The IoT devices get registered to the sensor monitoring platform maintained

Figure 5.13: Photograph of "cube" sensors used in the Gravity project to detect SLA violations and log them to the HIDALS system.

by an external sensor monitoring company. In addition, the hashed data storage pattern stores the SLAs in the distributed ledger's state database. Afterward, the handover phase begins. When a carrier receives the request for a handover, the carrier utilizes the distributed ledger to verify any SLA violations for that parcel. If the carrier decides to accept the parcel, the handover is logged. Therefore, a smart contract modifies the state information of the parcel. Handovers need to be signed by the current and the future holder of the parcel. The smart contract updates the state information to make it track the parcel's current holder. While the carrier delivers the parcel to its next destination, the carrier is not required to report incidents by themselves anymore. Instead, the sensor monitoring organization that owns the sensor inside the parcel monitors the values received from the sensor. When the sensor detects that the parcel got too warm and violates an SLA, the sensor monitoring organization creates a new transaction to the DLT. This transparent event log enables non-repudiation of SLA violations.

Once the parcel reaches its final destination, the receiver accepts or rejects the parcel. Additionally, the receiver signals the monitoring company to stop the monitoring process. The sensor gets detached from the parcel. This marks the end of the process.

The TAPE method enables to *compare* the initial and the improved process. Figure 5.5 shows this comparison based on the metrics after altering the process and adding the DLT trust patterns. Introducing new business logic into the process also added more process elements. These process elements themselves can add potential uncertainties. This phenomenon can be observed with the global uncertainty metric. Without applying any trust personas, the global uncertainty increased from 191 to 362 (+ 171). The global uncertainty metric is strongly correlated to the number of process elements in the process. Thus, when a process engineer wants to observe the *impact* that trust mitigation actions had on the process, not the GU metric but the *relevant uncertainty (RU)* metric needs to be observed from the viewpoint of a specific trust persona.

Within the Gravity project, the receiver trust persona fully trusts the sensor

		Initial **General**	Initial **Rec. TP**	After **General**	After **Rec. TP**	Change **General**	Change **Rec. TP**
Global	GU	**191.00**	191.00	**362.00**	362.00	**171.00**	171.00
	AEU	3.60	3.60	3.73	3.73	0.13	0.13
	UU	0.00	36.00	0.00	212.00	0.00	176.00
	RU	191.00	**155.00**	362.00	**150.00**	171.00	**-5.00**
Reg. Sender	LU	36.00	0.00	52	0	16.00	0.00
	RLU	0.19	0.00	0.144	0	-0.04	0.00
	LUB	-0.06	-0.33	-0.023	-0.2	0.04	0.13
	DI	2.00	2.00	3	3	1.00	1.00
	DD	0.00	0.00	0	0	0.00	0.00
	MI	1.00	1.00	3	3	2.00	2.00
	MD	0.00	0.00	0	0	0.00	0.00
Reg. Carrier M	LU	**100.00**	100.00	**98**	98	**-2.00**	-2.00
	RLU	**0.52**	1.00	**0.271**	1	**-0.25**	0.00
	LUB	**0.27**	0.67	**0.104**	0.8	**-0.17**	0.13
	DI	0.00	0.00	2	2	2.00	2.00
	DD	1.00	1.00	1	1	0.00	0.00
	MI	2.00	2.00	3	3	1.00	1.00
	MD	2.00	2.00	3	3	1.00	1.00
Reg. Carrier N	LU	0.00	0.00	0.00	0.00	0.00	0.00
	RLU	0.00	0.00	0.00	0.00	0.00	0.00
	LUB	-0.25	-0.33	-0.17	-0.20	0.08	0.13
	DI	0.00	0.00	0.00	0.00	0.00	0.00
	DD	0.00	0.00	0.00	0.00	0.00	0.00
	MI	2.00	2.00	2.00	2.00	0.00	0.00
	MD	1.00	1.00	1.00	1.00	0.00	0.00
Reg. Receiver	LU	55.00	n.a.	62.00	n.a.	7.00	n.a.
	RLU	0.29	n.a.	0.17	n.a.	-0.12	n.a.
	LUB	0.04	n.a.	0.01	n.a.	-0.03	n.a.
	DI	0.00	n.a.	0.00	n.a.	0.00	n.a.
	DD	1.00	n.a.	1.00	n.a.	0.00	n.a.
	MI	0.00	n.a.	1.00	n.a.	1.00	n.a.
	MD	2.00	n.a.	2.00	n.a.	0.00	n.a.
Reg. Sensor Mon.	LU	n.a.	n.a.	58.00	0.00	n.a.	n.a.
	RLU	n.a.	n.a.	0.16	0	n.a.	n.a.
	LUB	n.a.	n.a.	-0.006	-0.2	n.a.	n.a.
	DI	n.a.	n.a.	0	0	n.a.	n.a.
	DD	n.a.	n.a.	1	1	n.a.	n.a.
	MI	n.a.	n.a.	1	1	n.a.	n.a.
	MD	n.a.	n.a.	2	2	n.a.	n.a.
Reg. DLT	LU	n.a.	n.a.	92	0	n.a.	n.a.
	RLU	n.a.	n.a.	0.254	0	n.a.	n.a.
	LUB	n.a.	n.a.	0.087	-0.2	n.a.	n.a.
	DI	n.a.	n.a.	0	0	n.a.	n.a.
	DD	n.a.	n.a.	2	2	n.a.	n.a.
	MI	n.a.	n.a.	1	1	n.a.	n.a.
	MD	n.a.	n.a.	3	3	n.a.	n.a.

Table 5.5: The tables shows the trust metrics from the general perspective and from the perspective of the receiver trust persona (receiver TP) regarding the global metrics, the sender, carrier m, carrier n, and the receiver in a comparison between the initial version and the mitigated version.

GU: global uncertainty, AEU: average element uncertainty, UU: uncritical uncertainty, RU: relevant uncertainty, LU: total lane uncertainty, RLU: relative lane uncertainty, LUB: lane uncertainty balance, DI: data influence, DD: data dependency, MI: message influence, MD: message dependency.

monitoring organization and the distributed ledger governance consortium. The reasoning behind this is structured as follows. The sensor monitoring organization provides monitoring of IoT devices and SLAs as a service. In general, there

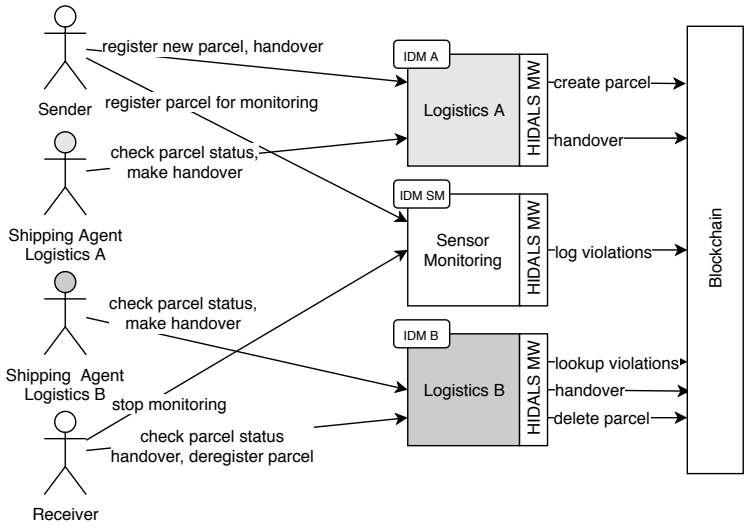

Figure 5.14: General use case and interactions between different actors in the HIDALS system.

are competitors for the sensor monitoring organization. The distributed ledger consortium consists of major logistics companies interested in providing a secure infrastructure to foster customer intimacy. These arguments were sufficient for the receiver trust persona in the Gravity project to trust both actors in the revised process. With these assumptions, the trust metrics show that the relevant uncertainty decreased by about 3% from 155 to 150.

The general approach of the HIDALS system within the Gravity project was to re-balance the process with decentralization. In the initial process, the carrier was responsible for a large portion of 100 uncertainties. This corresponds to 52% of all uncertainties in the process implying an imbalance of +0.27. In the revised process, uncertainty in the domain of the receiver was reduced by 2 to a total number of uncertainties of 98. Yet, after introducing the HIDLAS system, the carrier is only responsible for 27% of the uncertainties in the process. This comes from the fact that the process became more complex. Since the receiver trust persona trusts the DLT and the sensor monitoring organization, this metric illustrates the reduction of *relevant* uncertainties.

Applying TAPE Step 3: Implementing the trust-improved Process The implementation of the improved process with the HIDALS system utilizes different system components. Sender, receiver, and shipping agents use a mobile app for registering and handing over packages. A permissioned Hyperledger Fabric blockchain network handles all DLT transactions. Middleware services deployed on the premises of each logistics organization and sensor monitoring organization are the gateway to the DLT. Additionally, the middleware services serve as a point of integration towards legacy systems for ERP systems of the respective logistics organizations. The sensor monitoring organization handles the incoming data of autonomous sensors. It also maintains an instance of the HIDALS middleware service running on their premises as a gateway to persist SLA violations.

Figure 5.14 shows the interaction of different end-users with the system. They act as a sender, receiver, or shipping agent in a delivery process. Shipping agents of different logistics organizations interact with a mobile app (implemented in React Native[5]). The apps connect to the backend services of the logistics organizations and the sensor monitoring service provider. The mobile apps use endpoints to create new parcels with SLAs and delete parcels after they reached their final destinations. Additionally, functionality to request and accept a parcel handover is located at endpoints of the logistics organizations' backends. Towards the sensor monitoring platform, the mobile app has only endpoints to register a new parcel with defined SLAs for monitoring and to stop monitoring after the parcel has arrived at its final destination. All backend services utilize the local identity management systems for authentication and authorization.

The HIDALS middleware acts as a connector between the different logistics organizations' backend systems and the blockchain. Through that client, the logistics companies can evoke the `CreateParcel` and `DeleteParcel` transactions on behalf of the end consumers. The `HandoverFrom` and `HandoverTo` functions can be called on behalf of the end consumers or shipping agents. The transactions are signed with the private key of the logistics organization. The permissioned consortium governing the network and validation transactions consists of all involved logistics companies. Hyperledger Fabric [61] provides permission and role modeling. It can accommodate about 40000 transactions per second in certain setups [197]. This was identified as sufficient for the project. Internal business information, such as, information regarding handovers by end-consumers to logistics organizations (or vice versa), is stored on the premises of the logistics organization. They are also used in internal ERP systems. End consumers do not directly interact with the blockchain. Thus, their identity is obfuscated to third parties since the handover transactions are only signed with the keys of the related logistics organization. Intra-organizational handovers between shipping agents are not shown publicly. They are only saved in the internal systems of logistics companies. This mechanism protects the logistics organization from reverse-engineering internal processes through the blockchain logs by other participants in the collaborative process.

The mobile applications have interfaces towards the sensor monitoring platform for registering SLAs to a sensor. Additionally, they can also query the current status of a sensor. A sensor can have different states. It is either in the state "o.k." or "SLA violation occurred". When an SLA violation occurred, there is a description of the violation(s). Users that interact with the sensor monitoring endpoint use the authentication and authorization systems of the sensor monitoring organization. The HIDALS authorization concept enables end consumers to register new parcels with custom SLAs. Storing SLAs uses the logistics company's HIDLAS instance to transmit a transaction to the blockchain during the parcel registering process. Shipping agents of the carrier do not interact directly with the monitoring organization. The monitoring organization can evoke transactions to `CreateSensor`, `ChangeSensorStatus`, `DeleteSensor` as well as `CreateConditionViolation`. The sensor managing transactions are used to keep

[5] `https://facebook.github.io/react-native/`

track of the available sensors on the blockchain. Thus, an end consumer does not have the possibility to register a sensor to a parcel, which is not registered on the blockchain's state database. The status of a sensor can be either "o.k." or "defect". It is possible that a sensor can get defective during the delivery process. Hence, monitoring organizations have to log that information on the blockchain so that other participants of the process can judge the quality of the monitoring data. The sensors cubes within the parcels send their data through a NB-IoT connection to the monitoring organization. The data transmission is signed with the cube's private key. The public key of every sensor is stored in the DLT. If an SLA violation occurred, the responsible monitoring organization logs that occurrence immediately to the blockchain with a violation transaction. This transaction includes a hash of the data representing the violation incident. End-consumers can later retrieve the raw data from the sensor monitoring company and validate that the data is the original data from the sensor by checking the signature of the sensor and the hash on-chain. The possibility of bribery cannot be eliminated fully. Assuming the HIDALS system is used by different sensor monitoring companies who compete with each other, an organization would have no incentive to manipulate monitoring results. In that case, the end consumer would switch to another monitoring provider, and the monitoring organization would lose customers for future deliveries.

Discussion The case study of the Gravity project illustrated the application of TAPE to a complex real-world use case. This example has uncovered characteristics of the TAPE method regarding handling collapsed pools, monitoring changes through TAPE metrics, and context analysis. Additionally, the case study also illustrated the significance of DLTs for the mitigation of trust issues.

Handling and analyzing collapsed pools with Trust Mining is not intuitive. In Trust Mining's minimal viable artifact version, collapsed pools are handled as regular swimlanes without any process elements in them. Thus, collapsed pools count fully to calculate the relative lane uncertainties (RLU). Yet, they have no uncertainty in their own domain. This leads to a distortion of the RLU and LUB metrics. The origin of this distortion lies in the fact that TRUB does not distinguish between different types of swimlanes. Two different approaches can be utilized to address this issue. Firstly, process engineers may avoid using collapsed pools in their process. This approach is most practical for situations where collapsed pools are used to indicate multiple instances of the same type of actor in the process. The carrier m and carrier n pools in the analyzed process are examples for this. Leaving out the collapsed pool does not lead to any issues. The semantics that there can be multiple instances of the same type of actor in a process needs to be stated outside of the process diagram. In cases where collapsed pools are used to indicate interfaces to other organizations with business logic completely hidden, the first approach may not be practical. Therefore, a second approach to counteract this issue could exclude the collapsed pool from the lane-specific metrics.

Analyzing the impact of certain trust patterns in the process is possible by comparing the initial with the improved trust metrics. With this comparison, it

is possible to identify that introducing a DLT and another organization for sensor monitoring reduces the relevant uncertainty to the receiver by 5. Relatively, this leads to a 2% change. A 2% improvement seems rather insignificant at first sight. Interviews with the stakeholders in the Gravity project have shown that the changes pose a more drastic improvement to the initial process than the 2% metric might suggest. Customers of the partner involved in the Gravity project highly value the possibility of getting more insights into the process in a trusted way. Future work should focus on translating the measured improvement of uncertainties (the 2% measure) to the *perceived impact* of the process change.

The case study has uncovered context factors and game-theoretic principles that are important to trust persona but not yet covered in TAPE. For example, the receiver trusts the sensor monitoring company for assessing SLA violations correctly. This trust comes form the receiver assuming that the company knows that there are competitors that could supply the same services. Thus, the receiver believes that the sensor monitoring company has no *incentive* to manipulate triggering SLA violations consciously. This assumption is crucial for assessing the receiver trust policies and measuring the impact of the trust patterns. Yet, this type of reasoning regarding the *context* of the collaborative process is non-trivial. Future work may use game-theoretic approaches or social networks further to assess the best trust policies for certain trust personas.

5.4.2 PDP

In today's medical research process, the acquisition of highly sensitive personal data from patients and study participants is a central task. Yet, privacy and adherence to data processing regulations is a significant trust issue for patients [198]. The Patient Data Portal (PDP) research project proposes a novel privacy-centric knowledge generation process in medical research. Therefore, PDP uses distributed ledgers and trusted execution environments (TEEs). The following sections showcase the trust analysis of the processes behind the current medical research process (As-Is) and compare it to the altered process with PDP. The process was extracted from a real-world example in amyotrophic lateral sclerosis (ALS) research. Medical researchers at Hospital de Santa Maria, Faculdade de Medicina da Universidade de Lisboa in Lisbon, Portugal provided descriptions as the base of the As-Is process.

As-Is Process and Trust Issues The current empirical medical research processes consist of data collection, data analysis, and interpretation. The concrete process flows differ from case to case. In many cases, researchers are at the same time the practicing doctors of the patients. In this case, the doctor's clinic stores all patient-related data. When a new research study needs to acquire data, the researchers ask for the patient's permission to use the data. Since the data is already in the domain of the researcher, the acquisition process is simple. In other cases, different clinicians exchange data of their patients for studies. Therefore, they need the patient's consent. In the EU, there are legal regulations and ethical guidelines in place to which the researchers need to comply.

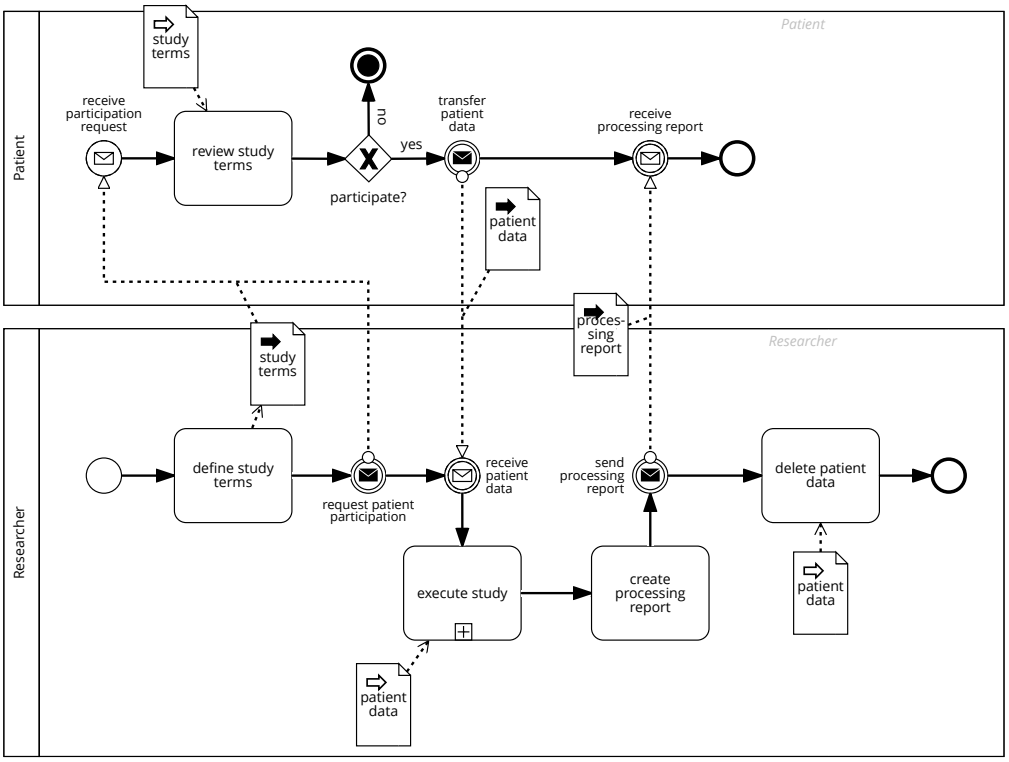

Figure 5.15: As-Is process of medical research and traditional data donations.

The PDP project aims to address a situation in which the patient physically possesses its patient data. Such situations frequently exist in medical research. For example, for studying the long-term effects of vaccines, patients are requested to have a personal "diary" to log their experienced vaccination reactions. Later, pharmaceutical researchers collect these datasets for empiric studies on common side effects of vaccines. This use case has gained attention during the coronavirus pandemic in 2021. Apps like SafeVac App[6] developed by the German Paul-Ehrlich-Institut are used to track possible vaccination reactions of COVID-19 vaccinations. These datasets are valuable for the assessment of vaccine safety. The same practice is common in situations where patients need to keep track of their long-term disease progression. For instance, amyotrophic lateral sclerosis (ALS) is a neurodegenerative disease that manifests in the progressive loss of motor neurons that control muscles. The disease encompasses a heterogeneous group of phenotypes with different progression rates. Most patients die from respiratory failure within three to five years. Yet, different progression patterns impact how patients can live their lives and the need for external assistance [199]. Current ALS research utilizes clinical data and data that the patients tracked periodically at home to identify these progression patterns. The patterns help create a custom disease prognosis and therapy plan.

The As-Is process describes a *data donation* uses case. A patient is in complete control of the own data, whereas the researcher needs to acquire it for clinical

[6]https://apps.apple.com/de/app/safevac/id1440303107

studies. The process is described in Figure 5.15. Initially, a researcher defines the terms of a study and its data processing. Therefore, the researcher has to adhere to a set of different regulations and ethical guidelines. The researcher requests the patient to participate in the study. Thus, the researcher has to transfer the study terms document to the patient. The patient reviews the study terms and decides whether to participate. If the patient chooses not to participate, the process terminates. In the positive case, the patient transfers the data to the researcher. The researcher collects the patient's data and executes the study. After the end of the study, the patient gets a report stating how the data has been used. Afterward, the researcher is required to delete the patient data.

In medical studies, many different patients need to donate their data. The depicted process illustrates the interaction between one patient and one researcher. Implicitly, it is possible to apply the process to any patient since it is always the same process.

	Initial **General**	Initial **Pat. TP**
Global Metrics		
GU	110.00	110.00
AEU	3.66	3.66
UU	0.00	68.0
RU	110.00	42.00
Reg. Researcher		
LU	62.00	42.00
RLU	0.56	1.00
LUB	0.06	0.00
DI	2.00	2.00
DD	1.00	1.00
MI	1.00	1.00
MD	1.00	1.00
Reg. Patient		
LU	48.00	n.a.
RLU	0.44	n.a.
LUB	-0.06	n.a.
DI	0.00	n.a.
DD	1.00	n.a.
MI	1.00	n.a.
MD	1.00	n.a.

Table 5.6: The tables shows the trust metrics from the general perspective and from the perspective of the patient trust persona (patient TP) regarding the global metrics and the patient and researcher lanes.

GU: global uncertainty, AEU: average element uncertainty, UU: uncritical uncertainty, RU: relevant uncertainty, LU: total lane uncertainty, RLU: relative lane uncertainty, LUB: lane uncertainty balance, DI: data influence, DD: data dependency, MI: message influence, MD: message dependency.

No.	uncertainty root	trust concern	process element	description
1	resource, software	integrity	activity	executing the study only in a way that no data is used for any other purpose
2	resource, software	integrity	activity	sending the correct data processing report that mirrors the real study

Table 5.7: Trust issues selected for mitigation in the PDP project.

Applying TAPE Step 1: Identifying Trust Issues Applying Trust Mining to the process leads to the trust-related metrics as shown in Table 5.6. Overall, the process has 110 uncertainties and an AEU of 3.66. The researcher is responsible for 62 uncertainties. This accounts for 56% of all uncertainties in the process (LUB +0.06). The patient is responsible for 48 (44%) of all uncertainties. The patient trust persona has trust policies that express trusting events in the process. The process has mostly message events for the interaction between the two different organizations. Thus, this trust policy expresses that the patient trusts the communication between them. With this trust policy applied, this leaves with 42 relevant uncertainties from the perspective of the patient trust persona.

Applying TAPE Step 2: Mitigating Trust Issues with DLT Trust Patterns The PDP project explored how uncertainties regarding the confidentiality trust concern in such medical research workflows can be minimized. Therefore, the project modified the process by introducing the following trust patterns:

1. **DLTs and TEEs**: The data analysis process is outsourced to trusted execution environments in a decentralized cloud. The data never reaches the researcher's domain of control.
2. **Decentralization**: The PDP decentralizes different parts of the workflow between actors. In the PDP workflow, researchers are only responsible for creating the data analysis algorithms. The cloud computing providers are in charge of executing them on the received data sets using TEEs. The researchers obtain the results, but never the raw data set.

Figure 5.16 illustrates the altered process. The process starts with the researcher defining the terms of study. These terms also include data processing agreements. Instead of directly sending the study terms to the test participant, the researcher creates an execution job through a transaction to the distributed ledger. The execution job defines the code of it and a job id. The job details get stored in the DLT's state database. After the creation of the job, the researcher requests the patient to participate in the study. The patient reviews the study terms and chooses to participate or to deny. In the negative case, the process terminates. In the positive case, the patient transfers the patient data to the TEE that the service worker hosts. Within the TEE, a job gets initialized, and a new transaction to the DLT signals that the chosen service worker is in charge of executing the job. After executing the job, the TEE creates an attestation of the

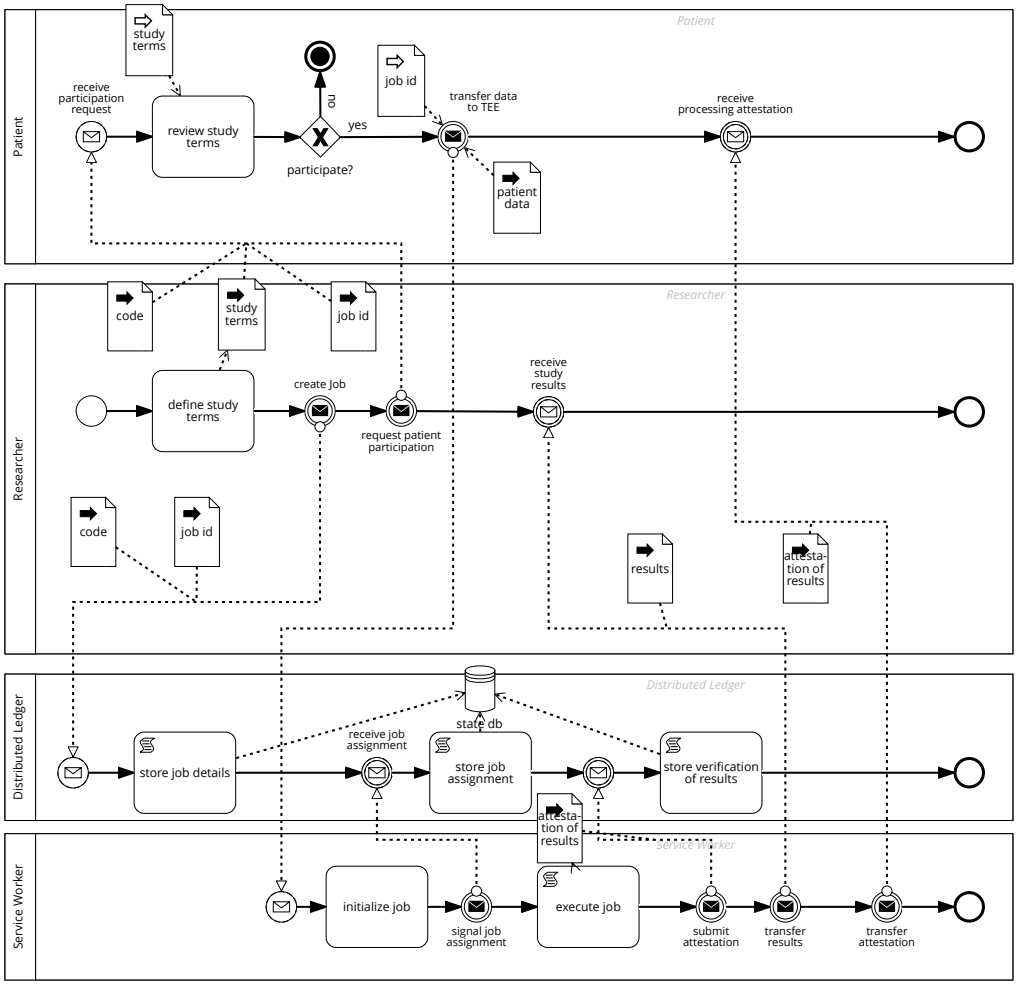

Figure 5.16: Trust-improved process of medical research data acquisition.

results. This attestation is a memory snapshot of the TEE's memory state after executing the code using the patient's data. The attestation is hashed and stored in the DLT's state database to guarantee its integrity and the non-repudiation of execution. Afterward, the TEE transfers the results of the execution to the researcher and the attestation of the execution to the patient.

In the altered process, the researcher defines the study terms and creates the code for data analysis. The researcher receives the results but never acquires control over the data. Also, the service worker never gains control over the data. The TEE is on a hardware level disconnected from the main processing unit of the worker. Within the TEE, there is a client that interfaces the blockchain. The TEE uses the information on the DLT to assign itself to the job. Only when the TEE is the worker for a certain job, the TEE can retrieve the data set. The TEE computes an attestation that proves that only the intended code has been executed using the intended data set. The patient never surrenders data control and can verify with the attestation proof and the code that the researcher conducted the promised job and nothing else.

The situation assumes that the patient trusts the implementation of the TEE

		Initial **General**	Initial **Pat. TP**	After **General**	After **Pat. TP**	Change **General**	Change **Pat TP**
Global	GU	110.00	110.00	201.00	201.00	91.00	91.00
	AEU	3.66	3.66	3.79	3.79	0.13	0.13
	UU	0.00	42.00	0.00	180.00	0.00	138.00
	RU	**110.00**	**68.00**	**201.00**	**21.00**	**91.00**	**-47.00**
Reg. Researcher	LU	**62.00**	**42.00**	**41.00**	**21.00**	**-21.00**	**-21.00**
	RLU	0.56	1.00	0.20	1.00	-0.36	0.00
	LUB	0.06	0.00	-0.05	0.67	-0.11	0.67
	DI	2.00	2.00	2.00	2.00	0.00	0.00
	DD	1.00	1.00	0.00	0.00	-1.00	-1.00
	MI	1.00	1.00	2.00	2.00	1.00	1.00
	MD	1.00	1.00	1.00	1.00	0.00	0.00
Reg. Patient	LU	48.00	n.a.	48.00	n.a.	n.a.	n.a.
	RLU	0.44	n.a.	0.24	n.a.	n.a.	n.a.
	LUB	-0.06	n.a.	-0.01	n.a.	n.a.	n.a.
	DI	0.00	n.a.	0.00	n.a.	n.a.	n.a.
	DD	1.00	n.a.	2.00	n.a.	n.a.	n.a.
	MI	1.00	n.a.	1.00	n.a.	n.a.	n.a.
	MD	1.00	n.a.	2.00	n.a.	n.a.	n.a.
Reg. DLT	LU	n.a.	n.a.	55.00	0.00	n.a.	n.a.
	RLU	n.a.	n.a.	0.27	0.00	n.a.	n.a.
	LUB	n.a.	n.a.	0.02	-0.33	n.a.	n.a.
	DI	n.a.	n.a.	0.00	0.00	n.a.	n.a.
	DD	n.a.	n.a.	0.00	0.00	n.a.	n.a.
	MI	n.a.	n.a.	0.00	0.00	n.a.	n.a.
	MD	n.a.	n.a.	2.00	2.00	n.a.	n.a.
Reg. Worker	LU	n.a.	n.a.	57.00	0.00	n.a.	n.a.
	RLU	n.a.	n.a.	0.28	0.00	n.a.	n.a.
	LUB	n.a.	n.a.	0.02	-0.33	n.a.	n.a.
	DI	n.a.	n.a.	0.00	0.00	n.a.	n.a.
	DD	n.a.	n.a.	0.00	0.00	n.a.	n.a.
	MI	n.a.	n.a.	3.00	3.00	n.a.	n.a.
	MD	n.a.	n.a.	1.00	1.00	n.a.	n.a.

Table 5.8: The tables shows the trust metrics from the general perspective and from the perspective of the patient trust persona (patient TP) regarding the global metrics, the researcher, the patient, the DLT, and the service worker in a comparison between the initial version and the mitigated version.

GU: global uncertainty, AEU: average element uncertainty, UU: uncritical uncertainty, RU: relevant uncertainty, LU: total lane uncertainty, RLU: relative lane uncertainty, LUB: lane uncertainty balance, DI: data influence, DD: data dependency, MI: message influence, MD: message dependency.

on a hardware level and the TEE's orchestration software. Additionally, the patient also has to trust the DLT governance consortium. This situation is reflected in trust policies added to the process. The trust policies state that the patient trusts the DLT and the service worker fully for all trust concerns in all tasks.

Figure 5.8 shows the changes in the trust-related metrics. In the initial process, 110 uncertainties were present. Out of these 110 uncertainties, 68 uncertainties were of relevance for the patient. In the altered process, 201 uncertainties can be observed. This is an increase of 91 uncertainties from the general perspective. The increase can be explained with the introduction of more process elements by adding the DLT and service workers. Yet, from the perspective of the patient trust persona, only 21 uncertainties are of relevance (-47 / -69%) in the improved process. This significant decrease results from the shift of the data processing tasks to the DLT and service worker. Since the patient trusts them, the trust issues in their domain are not relevant for the patient anymore.

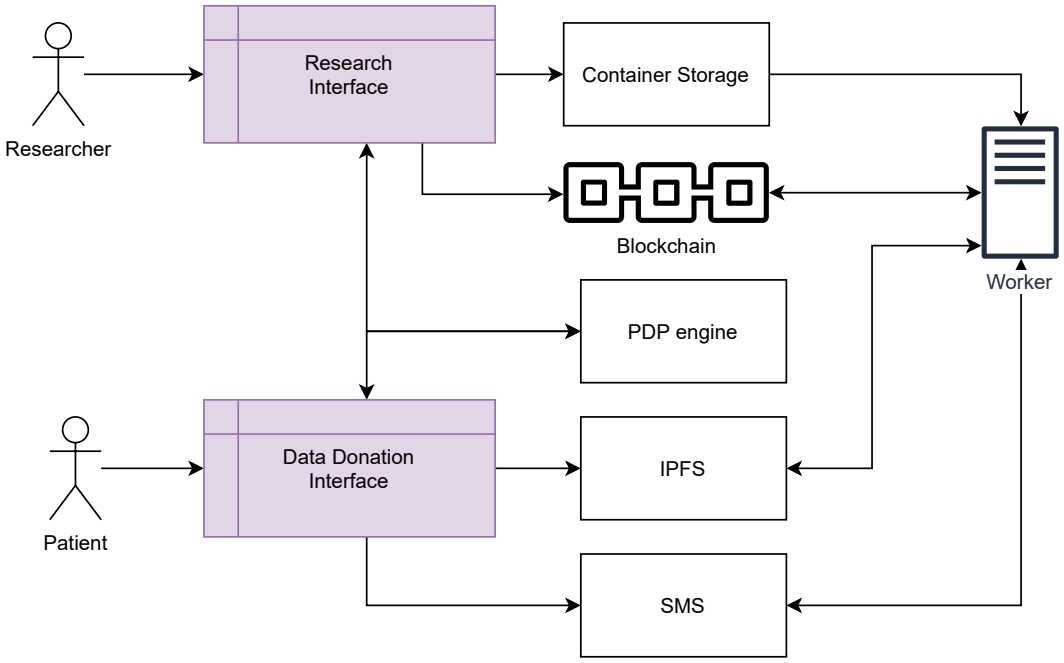

Figure 5.17: System architecture of the PDP system.

Applying TAPE Step 3: Implementing the trust-improved Process The PDP project implemented the proposed trust-improved process for medical research. The system architecture uses the iExec framework [200]. The framework facilitates scheduling of jobs and orchestrating them in TEEs using DLTs. The whole PDP system includes the components illustrated in Figure 5.17:

- **Researcher interface**: The researcher uses this web application to create new studies, request data donations, and facilitate code executions in TEEs using the iExec framework.
- **Data donation interface**: The patient uses this interface to supply patient data for research studies. Furthermore, the patient also discovers data donation requests with this interface.
- **PDP engine**: This centralized service is used for matchmaking between research studies and potential data donors. In general, there can be multiple PDP engines. The PDP engine does not store any medical data.
- **PDP toolkit**: This toolkit includes a set of software artifacts that aid the encryption and deployment of a data set in PDP.
- **Blockchain**: PDP uses a private Ethereum blockchain to store transactions for job scheduling and attestation. Therefore, the iExec SDK provides wrapper methods for rapid application development.
- **IPFS**: The interplanetary file system (IPFS) [201] stores (symmetrically) encrypted patient data. The service worker instance's TEEs retrieve the data for execution jobs from IPFS.

- **Container storage**: The container storage keeps version-controlled data analysis code in containers. The workers utilize this container storage to pull images for job execution.
- **Secret management service (SMS)**: The SMS is a TEE that manages symmetric keys. The workers use it to decrypt patient data. The software of the SMS is open-source so that every user can validate its trustworthiness.
- **Service worker instance**: The service worker instance executes the scheduled job in a TEE.

The iExec framework enables *decentralized cloud computing* [200]. In iExec, different parties can offer their computing resources in worker pools. Instances within worker pools are called service workers. Within the PDP project, service workers are always TEEs, even though iExec can be used for regular instances too. Service workers receive orders to execute. These orders are stored in an *order book* smart contract on a public Ethereum-based blockchain. The order book stores the hashes of the data set, job script, and the address of the assigned worker. This mechanism aims to ensure integrity and correct authorization. Therefore, also data sets and code for applications are hashed and stored in smart contract registries.

The implementation of the PDP system follows the high-level process as discussed in the previous section. The interactions with the system can be divided into four phases: the study initialization, data registry, donation requests, and execution.

In the *initialization* phase of the PDP workflow, a researcher defines the study terms and creates a single data analysis script. The analysis script must be designed in a way that it utilizes the data set of a *single patient* to generate insights. The researcher needs to create a container (using Docker[7]) capable of handling the necessary setup to execute the analysis task. The container gets compiled to an image and deployed to a container registry, such as Docker hub. The researcher creates a transaction to a smart contract on the blockchain that stores the application's hash. The order book is a smart contract of the iExec framework and will, in later stages, be utilized to assure integrity. After the creation of the application on-chain, the researcher registers the study in the PDP engine. All interactions with the blockchain and with the PDP engine are implemented using the researcher web interface.

In PDP, patients *register the data* they can provide for a research study with the PDP engine. Thus, patients have first to collect their patient data on their local machines and encrypt them. The *PDP toolkit* provides an application that supports symmetric encryption. The patients store their patient data on IPFS using the PDP toolkit. The keys to the encrypted data set get stored in the secret management system (SMS). The SMS is a service that runs in a TEE. The service is in charge of providing the keys to the data sets to the workers that execute tasks. Therefore, the SMS validates the authorization with the job information that is stored on-chain. The code of the SMS itself is open-source and can be validated by any actor. After deploying the data set to IPFS and the key to the

[7] https://www.docker.com

SMS, the patient registers a hash of the data set through a transaction in the blockchain's state database. This interaction is part of the iExec framework and aims to ensure the integrity of the data set. The deployment of the data set and transfer of keys is facilitated with the patient interface. After this transaction, the patient uses the interface to signal the availability of the data set for research studies to the PDP engine.

The researcher uses the researcher interface to make a *data donation request* to the PDP engine. The PDP engine searches for users that registered a data set and suggests potential candidates for the study. The researcher receives a list of available patient data sets. Based on this list, the researcher makes requests for using the patient data. The PDP engine forwards these requests to the patient interface. The patient has to decide whether to contribute data or not. In case the patient chooses to donate data, a transaction to an *order book* smart contract that is a part of the iExec framework is made. The order book keeps track of which application is authorized to use certain data sets. The authorization transaction is signed with the signature of the patient.

The *execution phase* of the research study brings together the researcher's code and the patient's data in a TEE. Therefore, the researcher uses the interface to provision a new TEE worker. The TEE worker verifies the authorization of the app to use the data set with the order book on-chain. Afterward, the TEE worker retrieves the encrypted data set from IPFS. The TEE worker requests the key to the data set from the SMS. On the other hand, the SMS verifies the order book and validates if the worker has been scheduled to execute an application. After the validation, the SMS returns the key to the worker. In addition, the TEE needs to retrieve the researcher's analysis code image from the container registry. The TEE is now in possession of the data set and the code. With these ingredients, the TEE can execute the code and return the result to the researcher. The TEE is physically detached from the host OS. Thus, the owner of the worker node can never get control over the data set. Once the execution of the job finishes, the worker creates an *attestation of execution*. This attestation is a memory snapshot of the TEE at the end of the job. The hash of the attestation gets stored in the DLT's state database. A link in the order book enables actors who are in possession of the code and the dataset to replay the execution. By replaying the execution and comparing its hash to the hash stored in the DLT's state database, patients can detect malicious data processing activities.

A large research study with 100 participants includes 100 of these four-step interactions. This is caused by a limitation of the iExec framework. The framework can only consume one data set in a single order, as illustrated in Figure 5.18. This requirement limits the applicability of PDP a to research setup similar to *federated learning*. Situations where the reseacher must first combine all data sets first cannot be covered by the PDP.

Discussion The PDP system demonstrated how DLTs and TEEs could enable medical research in which researchers never fully control the patients' data. Its implementation has shown that PDP is only applicable when the data analysis

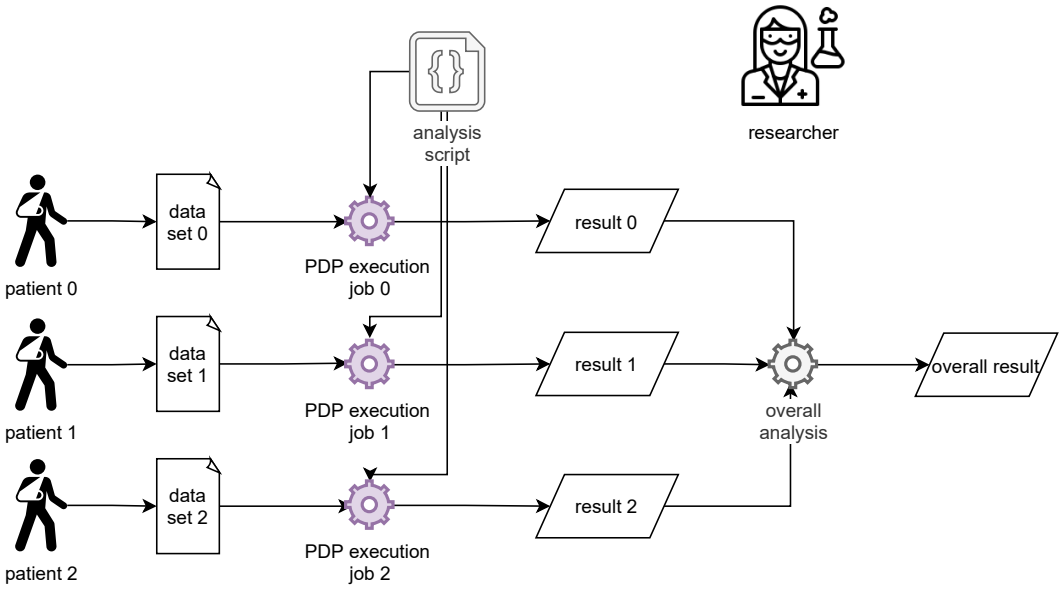

Figure 5.18: Illustration of the limitation of the PDP system. The data analysis must be dividable per patient. A overall analysis can only consume the outcomes of PDP execution tasks.

process can be meaningfully divided between different patients. This poses a threat to its generality.

Applying the TAPE method to this case study has illustrated the use of TAPE metrics for initial trust analysis, the use of trust patterns to mitigate trust issues, and the second application of Trust Mining to measure the impacts of the chosen trust patterns.

The initial process of patient data donations modeled two organizations. Implicitly the patient pool can be representative for n different patients that choose to donate their data. The business process model did not involve a collapsed pool. Thus, the distorting impact that the collapsed pools in the HIDALS case study exhibited could not be observed. The lane uncertainty balance in the PDP process was not biased with a swimlane that had no trust issues but accounted as a full swimlane.

The "smart contract TEE" trust pattern mitigated the uncertainty of the confidential handling of patient data in the PDP process. Yet, the trust pattern has a set of trust-enhancing capabilities that cannot be sufficiently modeled on the abstraction level of a business process. Different signatures, blockchain transactions, and the TEE's security mechanisms are additional *context* that needs to be taken into account when analyzing trust-related concepts of a process. Here, an additional context property on the levels of uncertainties could provide another benefit. With a context property, it is possible to mark an activity with "is executed in a TEE". The marking can be used together with a trust policy "patient trusts TEEs" to provide more *technical trust semantics*.

Assessing the impact of the applied trust patterns uncovered a change in the level of centrality in the process. The introduction of a distributed ledger and different service workers in a decentralized cloud shifted process elements away from the researcher. Thus, also the uncertainties associated with these process

elements changed their responsible organization. The LU and LUB metrics represented this shift understandably. Observing the global trust metrics regarding the relevant uncertainties to the patient showed a significant improvement of about 60%. In the HIDALS process, only a trust improvement of 3% could be observed. Hence, comparing the impact of trust patterns across different process models is challenging. The HIDALS and PDP processes were initially of different complexity and modeled on different abstraction levels. This leads to challenges in cross-process comparability. However, the case study showed that comparability between the initial and the mitigated version of a process is given. A measurable improvement of 60% is a significant change and a helpful insight for a process engineer.

5.5 Summary

This chapter evaluated the design science artifacts introduced in Chapter 3 and 4 with a variety of different methods.

- The conceptual-methodological evaluation in Section 5.1 discussed the limitations of TAPE. A demonstration of the TAPE metrics on a set of 137 example process models examined the discriminatory value of the metrics. The analysis uncovered a widespread distribution for the majority of the metrics. Thus, this leads to an interpretation of a high utility.
- Section 5.2 conducted controlled experiments and interviews with experts. Researchers guided the test participants in a hands-on process through the TAPE method. The final interview assessed the perceived usefulness and ease of use of TAPE and Trust Studio. The results have shown that all test participants experienced benefits in analyzing trust-related concepts with TAPE and Trust Studio. All participants could improve a trust issue in a given example process, regardless of their initial knowledge. The interviews uncovered action points for future work.
- The performance evaluation in Section 5.3 showed that executing the automated trust analysis is fast enough that their users can retrieve the results in less than a second.
- The two case studies in Section 5.4 demonstrated the introduced artifacts on real-world processes end-to-end. The TAPE method helped to identify trust issues and mitigate them with distributed ledgers. The implementation of the new processes were validated in real-world scenarios. The TAPE metrics illustrated the impact of the trust-aware process reengineering.

Overall, this chapter concludes that a minimum viable artifact state for all scientific contributions of this thesis has been reached. The evaluation generated insights that may be utilized for future work.

6 Conclusion

This thesis proposed new design science artifacts to analyze and mitigate trust issues in collaborative business processes using distributed ledgers. The following sections conclude the discussions on the minimal viable artifacts, their impact, and future work.

6.1 Results

The TAPE method provides a framework to conceptualize and analyze trust in collaborative business processes. Distributed ledger trust patterns and their classification can be used to mitigate trust issues and enhance the trustworthiness of a process from different perspectives. The Trust Studio tool supports process engineers and analysts in applying TAPE and trust patterns to their processes.

Design science is an inherently iterative research paradigm [19]. Thus, the goal of the introduced scientific contributions was to create *minimal viable artifacts* that provide utility to its users. The evaluation chapter conducted conceptual analyses, controlled experiments, and case studies to assess this utility.

Test participants experienced the TAPE method and DLT trust patterns in an extensive hands-on session in the controlled experiments. The experiment results indicate that the vast majority of test subjects deemed the artifacts to be useful. The experiments also identified potential improvements in the usability of Trust Studio. The performance analysis showed that the computational effort for Trust Mining's automated trust analysis is acceptable for process models of typical complexity. The versatility to represent certain trust characteristics has been shown during the analysis of the introduced trust-related metrics. The conceptual-methodological analysis discussed conceptual challenges and limitations of TAPE in its minimal viable artifact state. Finally, the case studies applied the introduced concepts to real-world processes. These studies demonstrated how TAPE and DLT trust patterns could help to solve existing trust issues in novel ways. Thus, this section concludes that the minimal viable artifact state of the TAPE method, Trust Studio, and the DLT trust patterns have been reached.

6.2 Impact

The outcomes of this thesis are design science artifacts that have the potential to pose an impact on three major use cases.

Challenge type	Goal	Challenge
conceptual	improve accessibility	abstraction bias
conceptual	improve generality	context property
methodological	improve accessibility	education in “trust thinking”
content	improve generality	additional trust-patterns
tooling	improve usability	trust studio improvements

Table 6.1: Summary of future action points for research on TAPE and trust patterns.

The TAPE method can help the creation of standards. For example, for the standardization of exchanging and processing confidential medical data, an iterative approach utilizing TAPE may increase the standard definition quality. This application of TAPE works as follows. First, an initial process is proposed. Then, this process gets analyzed using TAPE. Based on the trust analysis, the standard definition workgroup may introduce modifications. Afterward, the analysis starts again. It is possible to iterate over these steps until all stakeholders are justified with the uncertainty metrics related to the standard.

The TAPE method can be used as a supporting tool in making business decisions. In situations where companies need to choose if they wish to start a new business collaboration with an unknown party, TAPE’s trust analysis may be useful. TAPE enables commerce businesses to find the trust issues in their supply chains or service providers to find trust issues in the subprocesses of their contractors. With a fine-granular assessment of the uncertainties in the process and the willingness to accept them, applying TAPE can build *confidence* in a process.

As a final use case, the TAPE method can be applied for the trust-aware re-design of processes. For example, online multi-sided platforms may use TAPE to understand their users’ trust issues from their perspective without being biased. The developers of dApps might use TAPE to understand trust issues of their customers. Trust is an essential concept in decentralized finance. Having trust-aware processes might be the catalyst for broader options of novel banking workflows with distributed ledgers.

6.3 Future Directions

The following discusses potential future research directions that build on the outcomes of this thesis. The future research challenges can be divided into conceptual, methodological, content, and tooling challenges. The goal of these challenges is either to improve generality, accessibility, or usability.

Conceptual Challenges Conceptual challenges aim to improve the generality and accessibility of TAPE. The most considerable challenges in this category are addressing the abstraction bias and adding a context property to the trust analysis. The Trust Mining approach introduced annotations of trust-related

concepts to the elements of a business process model. Process models are capable of representing different granularity levels. For instance, the introduction of activities that define an internal subprocess leads to a distortion in the trust-related metrics, as Section 5.1 discussed. Mitigating the abstraction bias may lead to better interpretability.

The minimal viable artifact version of TAPE enables to classify trust issues according to roots of uncertainty, trust concerns, and process elements. However, more specific semantics could provide additional benefits. For instance, a context property that describes that an activity "uses open source software" might impact the trust tolerance profile of different trust personas. With the context property extension, trust policies can express that a certain trust persona "trusts open source software". This conceptual improvement would enable richer semantics in the trust analysis process.

Methodological Challenges The controlled experiments identified a high educational entry barrier to the use of TAPE and trust patterns. Comprehending the trust model and steps needed to derive insights is a non-trivial methodological challenge. Thus, introducing novel training methods to educate users on TAPE could improve its accessibility. Trust-aware business process management is a novel view on inter-organizational business processes. It distinguishes itself from other approaches like risk-aware process management by providing an external view on uncertainties. This helps specifically in situations where organizations cannot access insights into the details of an implementation of a process under the control of another organization. The controlled experiments and case studies have shown that process engineers benefit from an approach of "trust thinking". Trust thinking can be described as a way to deal with business processes by first analyzing uncertainties and creating the whole process based on the underlying trust implications. Thus, trust thinking is a "trust first" paradigm in business process management.

Trust thinking may have significant impacts on existing fields. The last years have seen a rising academic and professional interest in decentralized applications (dApps). A dApp is an application that is mostly decentralized and relies on distributed ledger technologies. The current main focus of dApps lies in the field of decentralized finance. Conceptually, a dApp implements an inter-organizational business process. Applying the TAPE method to the construction of new dApps is a potential catalyst to broaden the application domains of dApps.

Apart from dApps using distributed ledger technologies, additional application domains could benefit from the proposed trust thinking approach. In traditional distributed systems, software security and performance attributes are often the basis of system and interaction design. Inter-organizational processes are often implemented as distributed and decentralized systems where different organizations own certain parts of the entire system. Trust thinking proposes elevating trust as a consideration in such systems to the same level as security or performance. Thus, not only technical aspects are considered but also political implications that result from the decentralized nature of such systems.

Content Challenges The past chapters on trust patterns focused primarily on trust patterns that include distributed ledgers. However, the classification of the trust-enhancing capabilities of certain trust patterns can be applied regardless of the used technology. Thus, the classification of trust patterns like "add a second human resource to supervise the task" or "automate manual task" is also possible. Compiling a comprehensive list of non-DLT trust patterns can be seen as a content extension challenge for future research. Additionally, over time new DLT trust patterns may arise. With the help of the introduced classification, it is possible to characterize their impact clearly.

Tooling Challenges The TAPE method is built on the premise that it is possible to automate the analysis of trust issues in inter-organizational processes partially. An automated approach reduces the possibility of human errors in the process. The controlled experiments in the evaluation chapter have identified room for improvement on the usability of Trust Studio. Especially functionality to compare the initial version and the trust-improved version of a process may enhance its usability.

Addressing these challenges can further enhance the utility of the artifacts introduced in this thesis. An iterative research paradigm may lead to a forthcoming evolution of trust-aware business process management as a research field. The contributions of this thesis act as one of the first foundations in the new field.

A BPMN Elements

The following section describes the set of BPMN elements that this thesis uses as a reference.

A.1 Events

Events express reaching relevant states of interest during the process execution. These situations require enactment. Generally, events in business processes can be partitioned into start events, intermediate events, and end events. Start events trigger processes or subprocesses and can be interrupting or non-interrupting. Intermediate events can occur during the process execution. BPMN divides them further into throwing events (initiating and event handling flow), catching events (responding to triggered event flows), as well as interrupting and non-interrupting boundary events. Boundary events are attached to activities and handle states of interest that occur during the executing of the activity. End events are a signal for the termination of the process.

Events can have different types with different semantics as seen in Figure A.1.

- **Message events** are the primary type of communication in BPMN diagrams. They are especially often used in collaboration diagrams. There, message flows are the only interaction between subprocesses carried out by different organizations in separate swimlanes.
- **Timer events** represent time-based events such as a specific point in time approaching or timeout events. For instance, in the apartment rental example in Figure 2.5, the time event is used to indicate that the first day of vacation has arrived.
- **Escalation events** illustrate a point where a workflow reached a state where an actor with higher responsibility in the organization needs to take over the process. It is often used with error events.
- **Conditional events** are used to react to changes in business conditions or business rules.
- **Link events** are shortcuts for sequence flows and indicate that the end of a process marks the start of another process.
- **Error events** indicate that an error state has been reached. They mark that a process either gets terminated due to an error or that an error-handling flow is initiated.

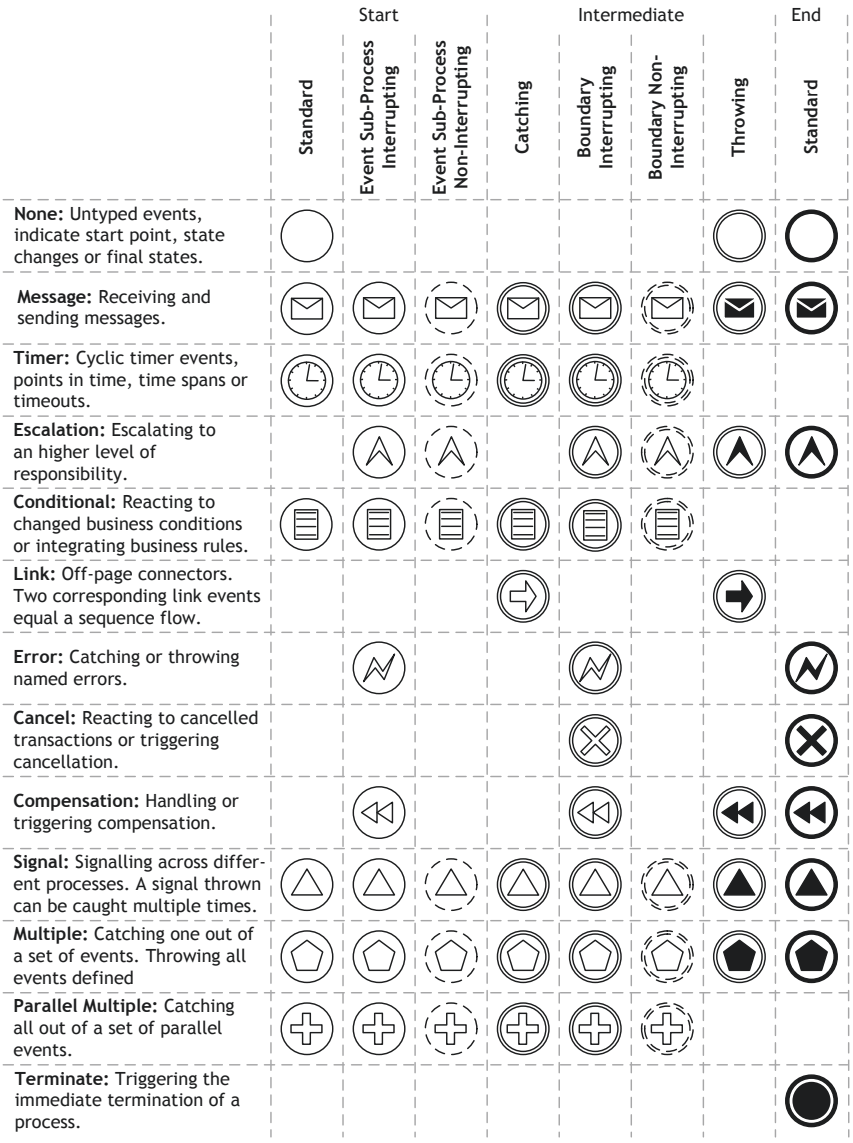

Figure A.1: Overview of BPMN 2.0 events. Modified excerpt from BPMN 2.0 poster (http://www.bpmb.de/images/BPMN2_0_Poster_EN.pdf)

- **Cancel events** are used to react to canceled transactions or trigger cancellation of transactions. Transactions will be discussed more in detail in the following paragraph.
- **Compensation events** express the compensation of a collaborator.
- **Signal events** are used to express communication between different processes. Thrown signal events can be caught multiple times.
- **Multiple events** can be caught and thrown through this element.
- **Parallel multiple events** are used to catch all events out of the set of parallel events.
- **Terminate events** represent the immediate termination of a process.

A.2 Activities

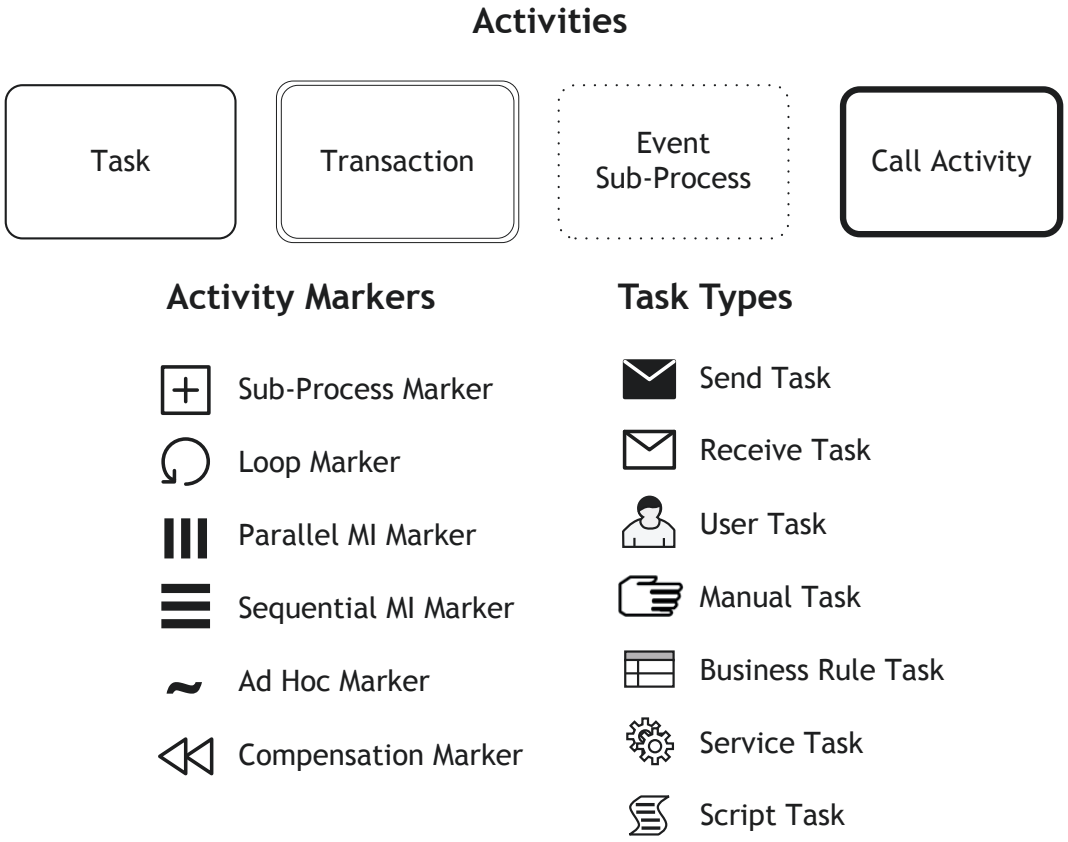

Figure A.2: Overview of BPMN 2.0 activities. Modified excerpt from BPMN 2.0 poster (http://www.bpmb.de/images/BPMN2_0_Poster_EN.pdf)

Activities represent units of work in business processes. In BPMN, activities can be nested so that every activity is either atomic or represents a *subprocess*. A subprocess is modeled by attaching a subprocess marker to an activity, as seen in Figure A.2. Atomic activities are also called *tasks*. BPMN 2.0 defines additional task types that describe the high-level properties of a task semantically. A *receive task* is a task that waits to receive a message and completes once the message arrives. *Send tasks* are defined analogously. *User tasks* represent tasks where users interact with a system to perform the task. For example, if a user uses a web app to look up how many items of a certain type are currently in stock, this is modeled as a user task. *Manual tasks* are executed purely by a human without interaction with a software system. Reading a letter or carrying a package are examples of manual tasks. *Service tasks* are executed using software through a service invocation. Using an API call to get the current weather information in Berlin is an example of a service task. *Script tasks* are executed through a script and directly executed by a business process engine. Similarly, *business rule tasks* are carrying decisions through business logic.

A *transaction* is a collection of activities that logically belong together. Activities grouped in a transaction are to be executed as a unit. Transaction protocols may be used to specify further how the activities belong together and how to

handle cases where one fails. An *event subprocess* is an activity that is placed into a process or subprocess. It gets activated with a start event and can either interrupt the higher-level process or run in parallel to it in a non-interrupting way. The last activity type BPMN defines is the *call activity*. This activity acts as a wrapper for a globally defined task or process and that is intended to be reused in different contexts. It is always marked with the subprocess marker symbol.

Apart from the already discussed subprocess marker, BPMN defines additional activity markers. The *loop marker* can be used to represent while or repeat-until loops. The *adhoc marker* represents a subprocess with unstructured parts. In addition, BPMN defines *multiple instance markers*. The *parallel multiple instance marker* indicates that multiple instances of the activity can be executed in parallel. In contrast, the *parallel multiple instance marker* represents the behavior for sequential execution analogously.

A.3 Sequence Flow

The BPMN standard calls control flow within an organization *sequence flow*. Solid arrows between flow objects, such as activities, events or gateways symbolize sequence flow (see Figure A.3). In addition to the normal sequence flow arrows, BPMN also provides capabilities to model *conditional flow* logic. Conditional flow arrows have a condition attached to them describing under which conditions this flow gets executed. It is possible to attach multiple condition flow arrows to a flow object modeling different conditions. If no condition is fulfilled, the *default flow* arrow defines the executed sequence flow in that case.

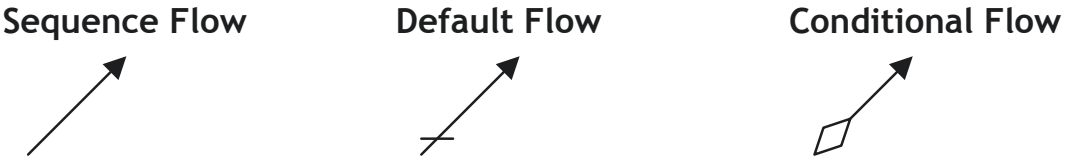

Figure A.3: Overview of BPMN 2.0 sequence flow. Modified excerpt from BPMN 2.0 poster (http://www.bpmb.de/images/BPMN2_0_Poster_EN.pdf)

A.4 Gateways

Gateways are in BPMN either *splits* or *joins* (see Figure A.4). *Exclusive gateways* route the sequence flow to exactly one of the outgoing branches. When merging back together, the exclusive gateway waits for one incoming branch to complete and triggers the outgoing flow. In contrast to the exclusive gateway, the *inclusive gateway* routes into one or more outgoing branches. When merging back together, the inclusive gateway requires all active incoming branches to complete. *Parallel gateways* are used to split sequence flow so that all outgoing branches are activated simultaneously. When merging back, the parallel gateway awaits all incoming branches and triggers the outgoing flow.

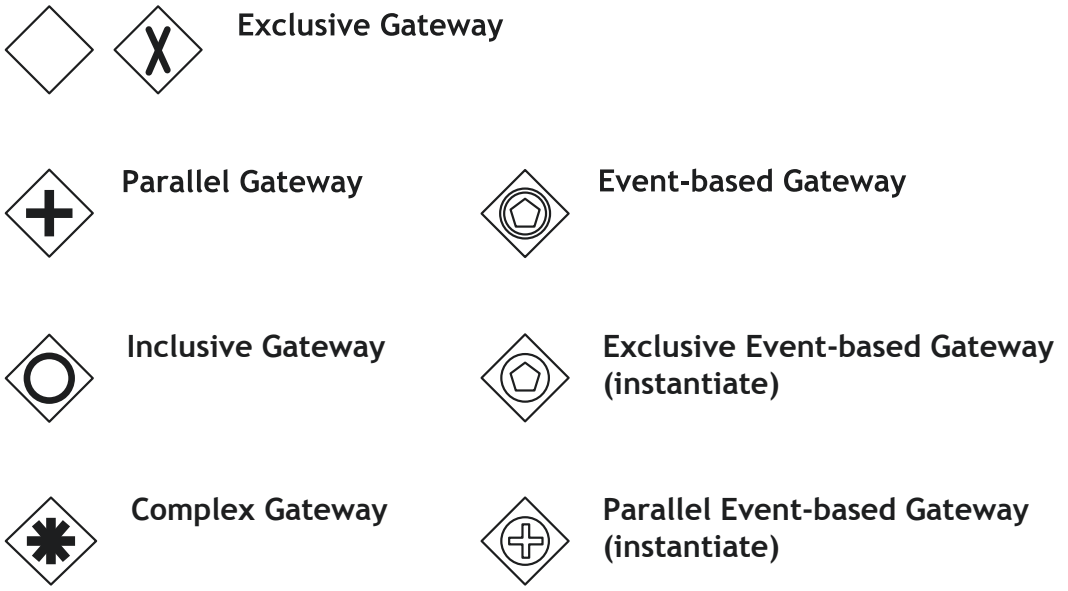

Figure A.4: Overview of BPMN 2.0 gateways. Modified excerpt from BPMN 2.0 poster (http://www.bpmb.de/images/BPMN2_0_Poster_EN.pdf)

In their default configuration, splits and joins are *data-based*. This means that the evaluation of the split and join criteria is always rooted in data associated with the process flow. In contrast to that, *event-based gateways* evaluate their logic based on the occurrence of events. Event-based gateways are always followed by catching events or receive tasks. The sequence flow is always routed to the event or task which happens first. Accordingly, there are variants for *exclusive event-based gateways* and *parallel event-based gateways*. In addition, BPMN specifies a *complex gateway* to express branching and merging behavior that is not captured by any other gateway.

A.5 Data

BPMN allows the modeling of data flows through artifacts called *data objects*. All data objects are linked through *data association* to flow objects or connecting objects. Besides basic data objects, *data input* and *data output* can be expressed with artifacts with arrows attached to them as seen in Figure A.5. The *collection data object* is a shortcut for a set of different data objects and the *data store object* models interaction with storage of data, such as databases.

A.6 Swimlanes

BPMN has a rich set of different tools to model inter- and intraorganizational collaboration. The primary elements to use for that are *swimlanes*. Swimlanes are either *pools*, which represent organizations or *lanes*, which represent different organizational units within one organization. Different organizations interact exclusively via *message flows* with each other, while units within one organization can interact through sequence flows. Figure A.6 shows an example from the

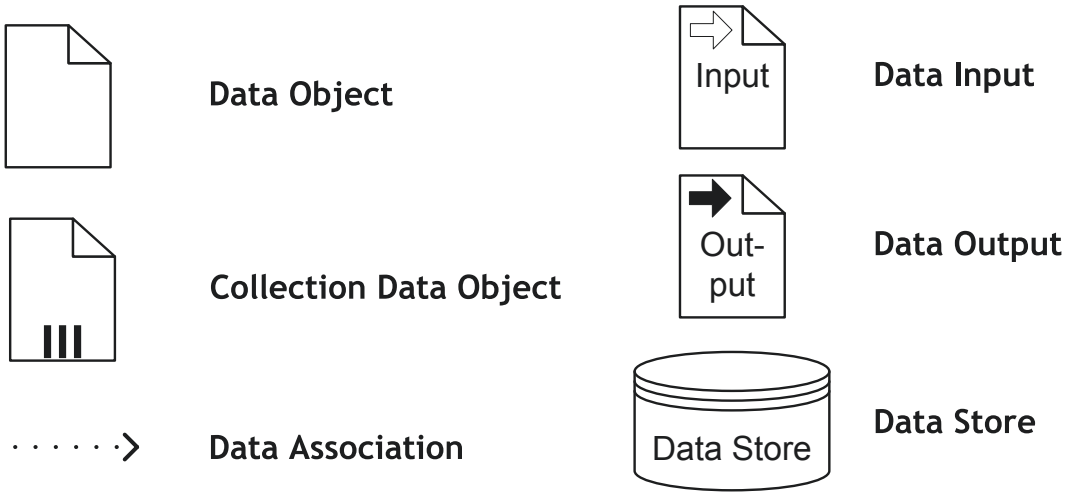

Figure A.5: Overview of BPMN 2.0 data objects. Modified excerpt from BPMN 2.0 poster (http://www.bpmb.de/images/BPMN2_0_Poster_EN.pdf)

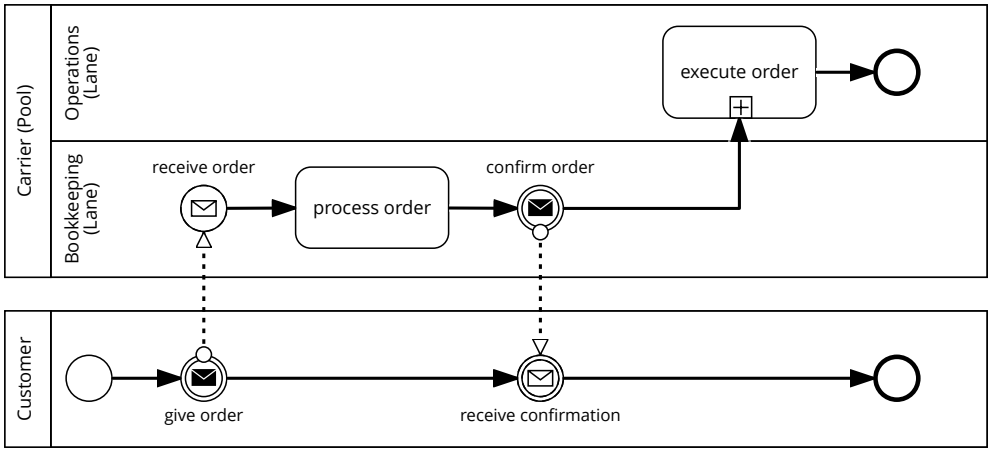

Figure A.6: Overview of BPMN 2.0 swimlanes. Modified excerpt from BPMN 2.0 poster (http://www.bpmb.de/images/BPMN2_0_Poster_EN.pdf)

logistics domain. A customer (pool) gives an order to deliver some items to the bookkeeping department (lane) of a carrier through a message. The message flow is indicated with the dotted arrow. After processing and confirming the order, the bookkeeping department forwards the order for executing to the operations department (lane). This is done through sequence flow since both lanes are in the same pool.

A.7 Conversations, Choreographies and Collaborations

BPMN defines different diagram types. *Conversation diagrams* model solely the interaction between different users within a process. *Choreography diagrams* model interaction and business logic and focus on the inter-organizational part aspect and message exchanges. For the remainder of this work, the primarily

used diagram type is a *collaboration diagram* as seen Figure A.6. Collaboration diagrams are the most commonly used diagram type and widely accepted both in industry and in research. Hence, choreographies and conversations will not be discussed further. A more detailed explanation of these diagram types can be found in [202].

B Controlled Experiment Script

This appendix describes the script used for the controlled experiments. All ten participants were involved in the same workflow.

B.1 Onboarding

Welcome to our controlled experiments. In this experiment, we will evaluate the utility of our methodological framework (TAPE) through its implementation (Trust Studio) and compare it to other approaches. Therefore, we need you. Your goal is to:

- identify trust issues in a given process
- mitigate these trust issues to make the process more trustworthy

Therefore, we will guide you through an experiment and work with you in a step-by-step fashion. The the experiment, you need the following material (see Google Drive).

B.1.1 Introduction to Process Models

Imagine some colleague found an idea online where a small manufacturer wants to build auto-real-time translation headphones (similar to this[1]). The headphones have a microphone that captures voice, transfers it in real time to your phone, where a translating app is producing audio of the translated sentences. Imagine it like the Babel-fish in Hitchhiker's Guide to the Galaxy.

Manufacturer made a small prototype but for the real world test and production of the product, she needs money. Getting a factory in China to produce the micro-electronics is initially very expensive. Thus, the manufacturer is raising money in a crowdfunding fashion. She tells everybody of her friends, colleagues and social media followers that if they now contribute 300€ of an investment, they will get a pair of babel fish headphones, when she reaches her goal of 30000€. That is the minimum amount she needs to find a manufacturer.

This diagram illustrates the process you will be tasked to analyze regarding its trustworthiness properties.

<verbal description of the process model>

Question regarding the process model? *note down the questions and answers*

[1] https://www.kickstarter.com/projects/1333364907/gks-translating-earbuds-for-every-occasion-and-every-county_category

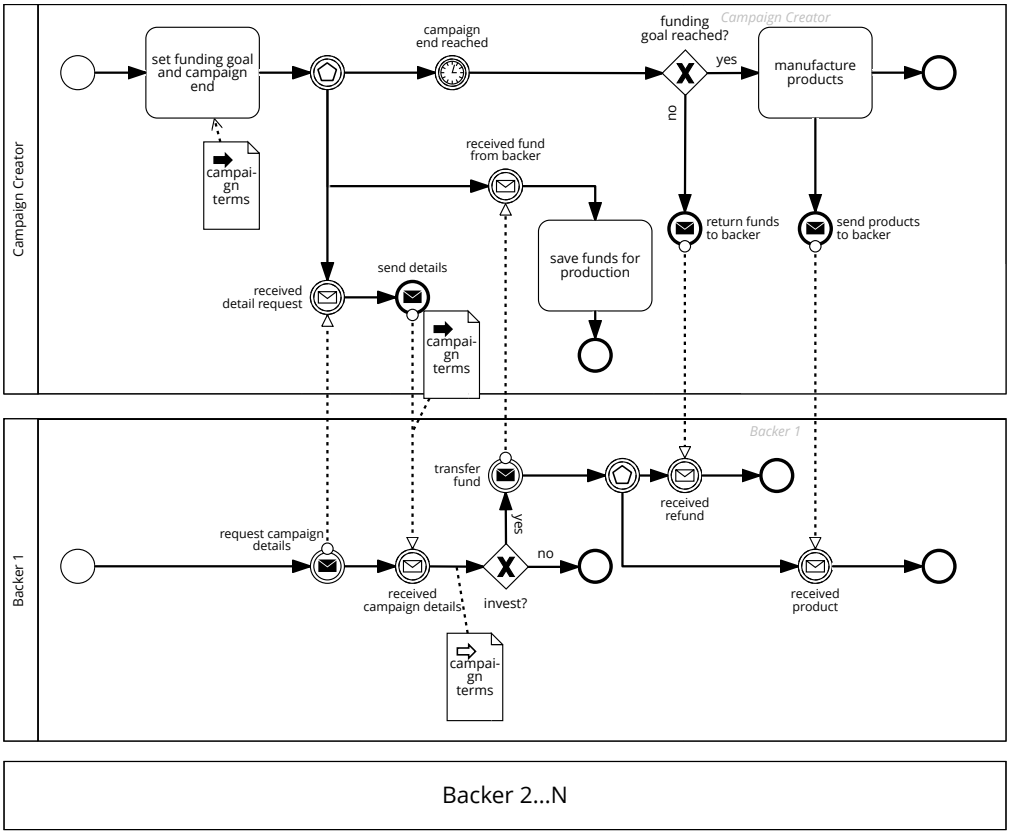

Figure B.1: Process for controlled experiment: peer-to-peer crowdfunding.

B.1.2 Introduction to Trust Concerns

Now you know what process you are dealing with. You will analyze the process model from the perspective of the space renter.

Therefore, we would like you to focus on the following trust concerns:

- integrity
- confidentiality
- availability
- non-repudiation
- performance
- resilience

In Trust Studio, these trust concerns get associated with parts of the process model that we were talking about. Therefore, Trust Studio does an automatic annotation that you will later utilize to get a better understanding of the trust situations in your process.

Are you familiar with these concepts? *<note questions of the test subject >*

B.1.3 Goal Statement

The goal of you within this experiment is to

1. identify trust issues that are relevant from your trust perspective on the process
2. find solution approaches to mitigate these trust issues

Therefore, we will provide you with a method and a guideline on how to apply this method.

B.1.4 Method Introduction

You are going to use a 4-step-method to identify trust issues at first. We will give you an introduction to each step to perform.

B.2 Experiment Execution (Analysis Part)

For the execution of the analysis, we have here a tool that will help you assess the trust issues in your process. Therefore, we are using Trust Studio as a tool. You can reach Trust Studio in the following way:

```
   http://trust.snet.tu-berlin.de
Username:  study_participant
Password:  asd87g387rg
```

This link is only available for the duration of the study.

You can see a button that leads you to a tutorial. Click it. *<note the test subjects reactions>*

In the next steps, we will go with you through the tutorial that explains the concepts behind trust studio. *<lead the test subject through the trust studio tutorial until the subject reaches the dashboard>*

Now that you are familiar with Trust Studio, we want to utilize it for the process we initially talked about. Therefore, we have here the process model as an XML representation that you can upload to Trust Studio. Go to the Google Drive folder you received earlier and download the .bpmn file.

Please upload the provided BPMN file to Trust Studio and wait for the mining process. *<note the test subjects reactions>*

This was the first step in Trust Studio. You can see now in the background that Trust Studio "mines" uncertainties in the process.

Please click on the pen tool to observe the annotations of the process. *<note the test subjects reactions>*

You can see here that Trust Studio annotated uncertainties to the elements of the process. They are quite abstract. For example, you can see by clicking on the "save funds for production" activity that there are several uncertainties related to that activity.

- Integrity describes if the execution is done correctly. For example, does the backer really store the funds and does not steal it to buy a Ferrari.

- Confidentiality describes if only the intended people can see that you are in that meeting room. Maybe it is important to you that not everybody knows how you invest your money?
- Non-repudiation means that the campaign creator can later not deny that the funds were stored.
- Availability means that everything needed for the activity is there. For example, the website for only banking not being down.
- Performance describes different properties of the activity. If it takes the campaign creator weeks to deposit the money, you might wonder what he was doing in the mean time.
- Resilience means that there are fallbacks if something goes wrong in the activity.

If you want to observe statistics on the trust in your process, you can see metrics in a dashboard.

Please click on the graph icon to go to the dashboard. *<note the test subjects reactions>*

Here, you can see the uncertainty metrics. The upper left shows a view on the relationships regarding trust with respect to messages and data. The middle shows a global distribution of uncertainties. The histograms on the bottom show which types of uncertainties are present in which parts of the process.

<note the test subjects reactions>

To get some perspective into the trust analysis, we will use trust personas. A persona is basically somebody who has a specific trust tolerance profile. You are the Backer in the process.

Explain what your trust policies are? Who do you trust for what? Can you model that with trust policies *<note the test subjects answers>*

Now compare the graphs in the dashboard with other perspectives. Can you see how your perspective is influenced? *<note the test subjects answers>*

B.3 Experiment Execution (Constructive Part)

Now that you have identified which trust issues are there present in the process, you have now the task to mitigate them.

What are the most urgent issues for you regarding trust in the process? *<note issue>*

Maybe let us look into save funds for production activity. This activity is quite important because here is the point where a campaign creator could commit fraud.

How can you translate this issue to an uncertainty root, trust concern, and process element? *<note classification>*

Here is a list of methods to mitigate some trust issues. We call these trust patterns. In our last studies, we focused on how blockchain technology can be used in trust patterns. Therefore, we have a list of trust patterns with blockchain:

https://docs.google.com/spreadsheets/d/
1KBkUEEIgf5Qe1NAa3dYlwepY7zD29DvVOWE7BNkK2fA/edit?usp=sharing

Can you apply these patterns to mitigate a trust issue? *<note patterns>*

We asked you before what you think your personal most important issues are. Do you think you addressed the issue? *<note opinion>*

B.4 Interview

The interview follows the questions illustrated in Table 5.1.

Bibliography

[1] K. Butner. "The smarter supply chain of the future". In: *Strategy & Leadership* 38.1 (2010), pp. 22–31.

[2] K. C. Laudon, C. G. Traver, et al. *E-commerce: business, technology, society*. 2016.

[3] F. Mazzella, A. Sundararajan, V. B. D'Espous, and M Möhlmann. "How digital trust powers the sharing economy". In: *IESE Business Review* 26.5 (2016), pp. 24–31.

[4] J. Mendling, I. Weber, W. V. D. Aalst, J. V. Brocke, C. Cabanillas, F. Daniel, S. Debois, C. D. Ciccio, M. Dumas, S. Dustdar, et al. "Blockchains for business process management-challenges and opportunities". In: *ACM Transactions on Management Information Systems (TMIS)* 9.1 (2018), p. 4.

[5] X. Li, B. Meng, and Z. Wang. "Recent patterns of global production and GVC participation". In: *Global Value Chain Development Report 2019* (2019), p. 9.

[6] A. Orendorff. *Global ecommerce statistics & trends to launch beyond borders*. `https://www.shopify.com/enterprise/global-ecommerce-statistics`. (Accessed on 07/24/2020). 2019.

[7] Deloitte. *What is digital economy?* `https://www2.deloitte.com/mt/en/pages/technology/articles/mt-what-is-digital-economy.html`. (Accessed on 07/29/2020). 2020.

[8] International Telecommunication Union. "Overview of the internet of things". In: *Next generation networks – frameworks and functional architecture models* Y2060 (2012).

[9] M. Rosemann. "Trust-Aware Process Design". In: *International Conference on Business Process Management*. Springer. 2019, pp. 305–321.

[10] P. Sztompka. *Trust: A sociological theory*. Cambridge University Press, 1999.

[11] *2020 Edelman Trust Barometer*. `https://www.edelman.com/trustbarometer`. (Accessed on 07/29/2020). 2020.

[12] B. Kaiser. *Targeted: The Cambridge Analytica whistleblower's inside story of how big data, Trump, and Facebook broke democracy and how it can happen again*. HarperCollins, 2019.

[13] J. Ewing. *Wachstum über alles: Der VW-Skandal*. Norton, 2017.

[14] Edelman Deutschland. *Die Autoindustrie gewinnt Vertrauen und steht vor immensen Herausforderungen*. `https://www.edelman.de/research/autoindustrie-gewinnt-vertrauen-alles-gut-also`. (Accessed on 07/29/2020). 2017.

[15] S. Nakamoto. "Bitcoin: A peer-to-peer electronic cash system". In: (2008).

[16] G. Wood et al. "Ethereum: A secure decentralised generalised transaction ledger". In: *Ethereum project yellow paper* 151.2014 (2014), pp. 1–32.

[17] F. Casino, T. K. Dasaklis, and C. Patsakis. "A systematic literature review of blockchain-based applications: current status, classification and open issues". In: *Telematics and Informatics* 36 (2019), pp. 55–81.

[18] W. Viriyasitavat, L. Da Xu, Z. Bi, and V. Pungpapong. "Blockchain and internet of things for modern business process in digital economy—the state of the art". In: *IEEE Transactions on Computational Social Systems* 6.6 (2019), pp. 1420–1432.

[19] A. R. Hevner, S. T. March, J. Park, and S. Ram. "Design science in information systems research". In: *MIS quarterly* (2004), pp. 75–105.

[20] T. Grandison and M. Sloman. "A survey of trust in internet applications". In: *IEEE Communications Surveys & Tutorials* 3.4 (2000), pp. 2–16.

[21] W. Viriyasitavat and A. Martin. "A survey of trust in workflows and relevant contexts". In: *IEEE Communications Surveys & Tutorials* 14.3 (2011), pp. 911–940.

[22] D. Gambetta et al. "Can we trust trust". In: *Trust: Making and breaking cooperative relations* 13 (2000), pp. 213–237.

[23] R. C. Mayer, J. H. Davis, and F. D. Schoorman. "An integrative model of organizational trust". In: *Academy of management review* 20.3 (1995), pp. 709–734.

[24] L. Lau and B. Whyte. *Technological aspects of trust in virtual organisations*. University of Leeds, School of Computer Studies, 1998.

[25] S. Castaldo, K. Premazzi, and F. Zerbini. "The meaning (s) of trust. A content analysis on the diverse conceptualizations of trust in scholarly research on business relationships". In: *Journal of business ethics* 96.4 (2010), pp. 657–668.

[26] R. Audi. "Some dimensions of trust in business practices: From financial and product representation to licensure and voting". In: *Journal of Business Ethics* 80.1 (2008), pp. 97–102.

[27] F. Bidault, R José, S. H. Zanakis, and P. S. Ring. "Willingness to rely on trust in global business collaborations: Context vs. demography". In: *Journal of World Business* 53.3 (2018), pp. 373–391.

[28] D. Faems, M. Janssens, A. Madhok, and B. V. Looy. "Toward an integrative perspective on alliance governance: Connecting contract design, trust dynamics, and contract application". In: *Academy of management journal* 51.6 (2008), pp. 1053–1078.

[29] L. Poppo and T. Zenger. "Do formal contracts and relational governance function as substitutes or complements?" In: *Strategic management journal* 23.8 (2002), pp. 707–725.

[30] A. Zaheer and N. Venkatraman. "Relational governance as an interorganizational strategy: An empirical test of the role of trust in economic exchange". In: *Strategic management journal* 16.5 (1995), pp. 373–392.

[31] C. Castelfranchi and R. Falcone. "Trust and control: A dialectic link". In: *Applied Artificial Intelligence* 14.8 (2000), pp. 799–823.

[32] S. Ruohomaa and L. Kutvonen. "Trust management survey". In: *International Conference on Trust Management*. Springer. 2005, pp. 77–92.

[33] L. Mui, M. Mohtashemi, and A. Halberstadt. "A computational model of trust and reputation". In: *Proceedings of the 35th Annual Hawaii International Conference on System Sciences*. IEEE. 2002, pp. 2431–2439.

[34] R. Farmer and B. Glass. *Building web reputation systems*. " O'Reilly Media, Inc.", 2010.

[35] *Fake Amazon reviews 'being sold in bulk' online - BBC News*. `https://www.bbc.com/news/business-56069472`. (Accessed on 03/03/2021).

[36] *Massiver Anstieg an Fake-Reviews bei Amazon seit Pandemiebeginn - IT-Business - derStandard.de > Web*. `https://www.derstandard.de/story/2000121589165/massiver-anstieg-an-fake-reviews-bei-amazon-seit-pandemiebeginn`. (Accessed on 03/03/2021).

[37] M. Weske. "Business process management architectures". In: *Business Process Management*. Springer, 2012, pp. 333–371.

[38] Object Management Group. "Notation (BPMN) Version 2.0 (2011)". In: *Available on: http://www. omg. org/spec/BPMN/2.0* 2 (2011).

[39] M. Dumas, M. La Rosa, J. Mendling, H. A. Reijers, et al. *Fundamentals of business process management*. Vol. 1. Springer, 2013.

[40] W. Van Der Aalst. *Process mining: discovery, conformance and enhancement of business processes*. Vol. 2. Springer, 2011.

[41] R. Edelman. *Edelman trust barometer: Global report*. (Accessed on 07/29/2020). 2018.

[42] G. Tennant. *Six Sigma: SPC and TQM in manufacturing and services*. Gower Publishing, Ltd., 2001.

[43] N. G. Mohammadi and M. Heisel. "Patterns for identification of trust concerns and specification of trustworthiness requirements". In: *Proceedings of the 21st European Conference on Pattern Languages of Programs*. 2016, pp. 1–20.

[44] N. G. Mohammadi and M. Heisel. "Enhancing business process models with trustworthiness requirements". In: *IFIP International Conference on Trust Management*. Springer. 2016, pp. 33–51.

[45] M. Heravizadeh. "Quality-aware business process management". PhD thesis. Queensland University of Technology, 2009.

[46] S. Suriadi, B. Weiß, A. Winkelmann, A. H. ter Hofstede, M. Adams, R. Conforti, C. Fidge, M. La Rosa, C. Ouyang, A. Pika, et al. "Current research in risk-aware business process management—overview, comparison, and gap analysis". In: *Communications of the Association for Information Systems* 34.1 (2014), p. 52.

[47] World Bank Group. *Distributed Ledger Technology (DLT) and Blockchain*. http://documents1.worldbank.org/curated/en/177911513714062215/pdf/122140-WP-PUBLIC-Distributed-Ledger-Technology-and-Blockchain-Fintech-Notes.pdf. (Accessed on 09/04/2020). 2017.

[48] P. Tasca and C. J. Tessone. "Taxonomy of blockchain technologies. Principles of identification and classification". In: *arXiv preprint arXiv:1708.04872* (2017).

[49] S. Davidson, P. De Filippi, and J. Potts. "Disrupting governance: The new institutional economics of distributed ledger technology". In: *Available at SSRN 2811995* (2016).

[50] N. Hackius and M. Petersen. "Blockchain in logistics and supply chain: trick or treat?" In: *Digitalization in Supply Chain Management and Logistics: Smart and Digital Solutions for an Industry 4.0 Environment. Proceedings of the Hamburg International Conference of Logistics (HICL), Vol. 23*. Berlin: epubli GmbH. 2017, pp. 3–18.

[51] W. Viriyasitavat, L. Da Xu, Z. Bi, and A. Sapsomboon. "Blockchain-based business process management (BPM) framework for service composition in industry 4.0". In: *Journal of Intelligent Manufacturing* (2018), pp. 1–12.

[52] X. Xu, I. Weber, M. Staples, L. Zhu, J. Bosch, L. Bass, C. Pautasso, and P. Rimba. "A taxonomy of blockchain-based systems for architecture design". In: *2017 IEEE International Conference on Software Architecture (ICSA)*. IEEE. 2017, pp. 243–252.

[53] Z. Zheng, S. Xie, H.-N. Dai, X. Chen, and H. Wang. "Blockchain challenges and opportunities: A survey". In: *International Journal of Web and Grid Services* 14.4 (2018), pp. 352–375.

[54] M. Vukolić. "The quest for scalable blockchain fabric: Proof-of-work vs. BFT replication". In: *International workshop on open problems in network security*. Springer. 2015, pp. 112–125.

[55] Y. Sompolinsky and A. Zohar. "Accelerating bitcoin's transaction processing". In: *Fast money grows on trees, not chains* (2013).

[56] Y. Lewenberg, Y. Sompolinsky, and A. Zohar. "Inclusive block chain protocols". In: *International Conference on Financial Cryptography and Data Security*. Springer. 2015, pp. 528–547.

[57] *Instantly Move Money to All Corners of the World | Ripple*. https://ripple.com/. (Accessed on 09/07/2020).

[58] *Monax Website*. http://monax.io/. (Accessed on 09/07/2020).

[59] D. Khovratovich and J. Law. "Sovrin: digital identities in the blockchain era". In: *Github Commit by jasonalaw October* 17 (2017).

[60] JP Morgan Chase. *Quorum Whitepaper*. 2017. url: https://github.com/jpmorganchase/quorum/blob/master/docs/Quorum\%20Whitepaper\%20v0.2.pdf (visited on 02/10/2020).

[61] E. Androulaki, A. Barger, V. Bortnikov, C. Cachin, K. Christidis, A. De Caro, D. Enyeart, C. Ferris, G. Laventman, Y. Manevich, et al. "Hyperledger fabric: a distributed operating system for permissioned blockchains". In: *Proceedings of the thirteenth EuroSys conference*. 2018, pp. 1–15.

[62] G.-T. Nguyen and K. Kim. "A survey about consensus algorithms used in blockchain". In: *Journal of Information processing systems* 14.1 (2018), pp. 101–128.

[63] S. Wan, M. Li, G. Liu, and C. Wang. "Recent advances in consensus protocols for blockchain: a survey". In: *Wireless networks* 26.8 (2020), pp. 5579–5593.

[64] W. Wang, D. T. Hoang, P. Hu, Z. Xiong, D. Niyato, P. Wang, Y. Wen, and D. I. Kim. "A survey on consensus mechanisms and mining strategy management in blockchain networks". In: *IEEE Access* 7 (2019), pp. 22328–22370.

[65] L. S. Sankar, M Sindhu, and M Sethumadhavan. "Survey of consensus protocols on blockchain applications". In: *2017 4th International Conference on Advanced Computing and Communication Systems (ICACCS)*. IEEE. 2017, pp. 1–5.

[66] Y. Xiao, N. Zhang, W. Lou, and Y. T. Hou. "A survey of distributed consensus protocols for blockchain networks". In: *IEEE Communications Surveys & Tutorials* 22.2 (2020), pp. 1432–1465.

[67] J. Li, N. Li, J. Peng, H. Cui, and Z. Wu. "Energy consumption of cryptocurrency mining: A study of electricity consumption in mining cryptocurrencies". In: *Energy* 168 (2019), pp. 160–168.

[68] A. Back. "Hashcash-a denial of service counter-measure". In: (2002).

[69] Ethereum Wiki. *Ethhash*. `https://eth.wiki/en/concepts/ethash/ethash`. (Accessed on 03/04/2021).

[70] *Documentation of Peercoin Cryptocurrency*. `https://docs.peercoin.net/`. (Accessed on 03/05/2021).

[71] J. Kwon. "Tendermint: Consensus without mining". In: *Draft v. 0.6, fall* 1.11 (2014).

[72] V. Buterin and V. Griffith. "Casper the friendly finality gadget". In: *arXiv preprint arXiv:1710.09437* (2017).

[73] F. Schuh and D. Larimer. "Bitshares 2.0: general overview". In: *Available: http://docs.bitshares.org/downloads/bitshares-general.pdf* (2017). (Accessed on 03/05/2021).

[74] *Steem - An incentivized, blockchain-based, public content platform*. `https://steem.com/steem-whitepaper.pdf`. (Accessed on 03/05/2021). 2018.

[75] *Lisk - Blockchain Application Platform*. `https://lisk.io/`. (Accessed on 03/05/2021).

[76] A. Gervais, G. O. Karame, K. Wüst, V. Glykantzis, H. Ritzdorf, and S. Capkun. "On the security and performance of proof of work blockchains". In: *Proceedings of the 2016 ACM SIGSAC conference on computer and communications security*. 2016, pp. 3–16.

[77] L. Lamport, R. Shostak, and M. Pease. "The Byzantine generals problem". In: *Concurrency: the Works of Leslie Lamport*. 2019, pp. 203–226.

[78] D. Mazieres. "The Stellar Consensus Protocol". In: *A Federated Model for Internet-level Consensus. Version July* 14 (2015).

[79] G. Wood. *PoA Private Chains*. `https://github.com/ethereum/guide/blob/master/poa.md`. (Accessed on 03/09/2021).

[80] Parity Ethereum. *Proof-of-Authority Chains*. url: `https://wiki.parity.io/Proof-of-Authority-Chains`.

[81] D5000 (Githubuser). *What is Proof of Burn?* (Accessed on 03/09/2021). 2014. url: `https://github.com/slimcoin-project/Slimcoin/wiki`.

[82] P4Titan. *Slimcoin A Peer-to-Peer Crypto-Currency with Proof-of-Burn*. `https://slimcoin.info/whitepaperSLM.pdf`. (Accessed on 03/09/2021).

[83] Nem.io. *Technical Reference*. (Accessed on 03/09/2021). 2018. url: `https://nem.io/wp-content/themes/nem/files/NEM_techRef.pdf`.

[84] L. Page, S. Brin, R. Motwani, and T. Winograd. "The PageRank Citation Ranking: Bringing Order to the Web". In: *World Wide Web Internet And Web Information Systems* 54.1999-66 (1998), pp. 1–17. issn: 1752-0509. doi: `10.1.1.31.1768`. url: `http://ilpubs.stanford.edu:8090/422`.

[85] N. Szabo. "Smart contracts". In: *Nick Szabo's Papers and Concise Tutorials* (1994).

[86] A. M. Antonopoulos and G. Wood. *Mastering ethereum: building smart contracts and dapps*. O'Reilly Media, 2018.

[87] R. G. Brown, J. Carlyle, I. Grigg, and M. Hearn. "Corda: an introduction". In: *R3 CEV, August* 1 (2016), p. 15.

[88] *Polkadot: Decentralized Web 3.0 Blockchain Interoperability Platform*. `https://polkadot.network/`. (Accessed on 03/22/2021).

[89] P. Zheng, Z. Zheng, X. Luo, X. Chen, and X. Liu. "A detailed and real-time performance monitoring framework for blockchain systems". In: *2018 IEEE/ACM 40th International Conference on Software Engineering: Software Engineering in Practice Track (ICSE-SEIP)*. IEEE. 2018, pp. 134–143.

[90] S. Viswanathan and A. Shah. *The Scalability Trilemma in Blockchain*. `https://medium.com/@aakash_13214/the-scalability-trilemma-in-blockchain-75fb57f646df`. (Accessed on 03/09/2021). 2018.

[91] S. Gilbert and N. Lynch. "Brewer's conjecture and the feasibility of consistent, available, partition-tolerant web services". In: *Acm Sigact News* 33.2 (2002), pp. 51–59.

[92] J. Xie, F. R. Yu, T. Huang, R. Xie, J. Liu, and Y. Liu. "A survey on the scalability of blockchain systems". In: *IEEE Network* 33.5 (2019), pp. 166–173.

[93] *Bitcoin Cash*. `https://bitcoincash.org/`. (Accessed on 03/09/2021).

[94] *Bip 152 - The Compact Block Relay*. `https://github.com/bitcoin/bips/blob/master/bip-0152.mediawiki`. (Accessed on 03/09/2021).

[95] I. Eyal, A. E. Gencer, E. G. Sirer, and R. Van Renesse. "Bitcoin-ng: A scalable blockchain protocol". In: *13th {USENIX} symposium on networked systems design and implementation ({NSDI} 16)*. 2016, pp. 45–59.

[96] Y. Gilad, R. Hemo, S. Micali, G. Vlachos, and N. Zeldovich. "Algorand: Scaling byzantine agreements for cryptocurrencies". In: *Proceedings of the 26th Symposium on Operating Systems Principles*. 2017, pp. 51–68.

[97] A. Kiayias, A. Russell, B. David, and R. Oliynykov. "Ouroboros: A provably secure proof-of-stake blockchain protocol". In: *Annual International Cryptology Conference*. Springer. 2017, pp. 357–388.

[98] E. Kokoris-Kogias, P. Jovanovic, L. Gasser, N. Gailly, E. Syta, and B. Ford. "Omniledger: A secure, scale-out, decentralized ledger via sharding". In: *2018 IEEE Symposium on Security and Privacy (SP)*. IEEE. 2018, pp. 583–598.

[99] M. Zamani, M. Movahedi, and M. Raykova. "Rapidchain: Scaling blockchain via full sharding". In: *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*. 2018, pp. 931–948.

[100] J. Poon and T. Dryja. *The bitcoin lightning network: Scalable off-chain instant payments*. 2016.

[101] J. Coleman, L. Horne, and L. Xuanji. *Counterfactual: Generalized state channels*. 2018.

[102] J. Kwon and E. Buchman. *Cosmos Whitepaper*. `https://cosmos.network/resources/whitepaper`. (Accessed on 03/09/2021).

[103] M. Crosby, P. Pattanayak, S. Verma, V. Kalyanaraman, et al. "Blockchain technology: Beyond bitcoin". In: *Applied Innovation* 2.6-10 (2016), p. 71.

[104] L. Peng, W. Feng, Z. Yan, Y. Li, X. Zhou, and S. Shimizu. "Privacy preservation in permissionless blockchain: A survey". In: *Digital Communications and Networks* (2020).

[105] Z. Zheng, S. Xie, H. Dai, X. Chen, and H. Wang. "An overview of blockchain technology: Architecture, consensus, and future trends". In: *2017 IEEE international congress on big data (BigData congress)*. IEEE. 2017, pp. 557–564.

[106] D. Ron and A. Shamir. "Quantitative analysis of the full bitcoin transaction graph". In: *International Conference on Financial Cryptography and Data Security*. Springer. 2013, pp. 6–24.

[107] S. Meiklejohn, M. Pomarole, G. Jordan, K. Levchenko, D. McCoy, G. M. Voelker, and S. Savage. "A fistful of bitcoins: characterizing payments among men with no names". In: *Proceedings of the 2013 conference on Internet measurement conference*. 2013, pp. 127–140.

[108] Q. Feng, D. He, S. Zeadally, M. K. Khan, and N. Kumar. "A survey on privacy protection in blockchain system". In: *Journal of Network and Computer Applications* 126 (2019), pp. 45–58.

[109] M. Conti, E. S. Kumar, C. Lal, and S. Ruj. "A survey on security and privacy issues of bitcoin". In: *IEEE Communications Surveys & Tutorials* 20.4 (2018), pp. 3416–3452.

[110] R. Zhang, R. Xue, and L. Liu. "Security and privacy on blockchain". In: *ACM Computing Surveys (CSUR)* 52.3 (2019), pp. 1–34.

[111] H. Nakamura, K. Miyamoto, and M. Kudo. "Inter-organizational Business Processes Managed by Blockchain". In: *International Conference on Web Information Systems Engineering*. Springer. 2018, pp. 3–17.

[112] M. Swan. "Anticipating the economic benefits of blockchain". In: *Technology innovation management review* 7.10 (2017), pp. 6–13.

[113] D. Galvin. "Ibm and walmart: Blockchain for food safety". In: (2017).

[114] Y. Chen and C. Bellavitis. "Blockchain disruption and decentralized finance: The rise of decentralized business models". In: *Journal of Business Venturing Insights* 13 (2020), e00151.

[115] X. Xu, I. Weber, and M. Staples. *Architecture for blockchain applications*. Springer, 2019.

[116] X. Xu, C. Pautasso, L. Zhu, V. Gramoli, A. Ponomarev, A. B. Tran, and S. Chen. "The blockchain as a software connector". In: *2016 13th Working IEEE/IFIP Conference on Software Architecture (WICSA)*. IEEE. 2016, pp. 182–191.

[117] N. R. Mehta, N. Medvidovic, and S. Phadke. "Towards a taxonomy of software connectors". In: *Proceedings of the 22nd international conference on Software engineering*. 2000, pp. 178–187.

[118] P. Clements, D. Garlan, R. Little, R. Nord, and J. Stafford. "Documenting software architectures: views and beyond". In: *25th International Conference on Software Engineering, 2003. Proceedings.* IEEE. 2003, pp. 740–741.

[119] N. Medvidovic and R. N. Taylor. "Software architecture: foundations, theory, and practice". In: *2010 ACM/IEEE 32nd International Conference on Software Engineering*. Vol. 2. IEEE. 2010, pp. 471–472.

[120] R. Van Mölken. *Blockchain across Oracle: Understand the details and implications of the Blockchain for Oracle developers and customers*. Packt Publishing Ltd, 2018.

[121] M. Nofer, P. Gomber, O. Hinz, and D. Schiereck. "Blockchain". In: *Business & Information Systems Engineering* 59.3 (2017), pp. 183–187.

[122] I. Weber, X. Xu, R. Riveret, G. Governatori, A. Ponomarev, and J. Mendling. "Untrusted business process monitoring and execution using blockchain". In: *International Conference on Business Process Management*. Springer. 2016, pp. 329–347.

[123] J. A. Garcia-Garcia, N. Sánchez-Gómez, D. Lizcano, M. Escalona, and T. Wojdyński. "Using Blockchain to Improve Collaborative Business Process Management: Systematic Literature Review". In: *IEEE Access* 8 (2020), pp. 142312–142336.

[124] M. Müller and P. Ruppel. "Process Mining for Decentralized Applications". In: *2019 IEEE International Conference on Decentralized Applications and Infrastructures (DAPPCON)*. IEEE. 2019, pp. 164–169.

[125] C. Klinkmüller, A. Ponomarev, A. B. Tran, I. Weber, and W. van der Aalst. "Mining blockchain processes: Extracting process mining data from blockchain applications". In: *International Conference on Business Process Management*. Springer. 2019, pp. 71–86.

[126] Y. Cen, H. Wang, and X. Li. "Improving business process interoperability by shared ledgers". In: *Proceedings of the 6th International Conference on Informatics, Environment, Energy and Applications*. 2017, pp. 89–93.

[127] J. Ladleif, M. Weske, and I. Weber. "Modeling and enforcing blockchain-based choreographies". In: *International Conference on Business Process Management*. Springer. 2019, pp. 69–85.

[128] G. Falazi, M. Hahn, U. Breitenbücher, and F. Leymann. "Modeling and execution of blockchain-aware business processes". In: *SICS Software-Intensive Cyber-Physical Systems* 34.2 (2019), pp. 105–116.

[129] C. Di Ciccio, A. Cecconi, J. Mendling, D. Felix, D. Haas, D. Lilek, F. Riel, A. Rumpl, and P. Uhlig. "Blockchain-based traceability of inter-organisational business processes". In: *International Symposium on Business Modeling and Software Design*. Springer. 2018, pp. 56–68.

[130] O. López-Pintado, L. García-Bañuelos, M. Dumas, and I. Weber. "Caterpillar: A Blockchain-Based Business Process Management System." In: *BPM (Demos)*. 2017.

[131] O. López-Pintado, L. García-Bañuelos, M. Dumas, I. Weber, and A. Ponomarev. "Caterpillar: A business process execution engine on the Ethereum blockchain". In: *Software: Practice and Experience* 49.7 (2019), pp. 1162–1193.

[132] O. López-Pintado, M. Dumas, L. García-Bañuelos, and I. Weber. "Dynamic role binding in blockchain-based collaborative business processes". In: *International Conference on Advanced Information Systems Engineering*. Springer. 2019, pp. 399–414.

[133] L. Mercenne, K.-L. Brousmiche, and E. B. Hamida. "Blockchain Studio: A Role-Based Business Workflows Management System". In: *2018 IEEE 9th Annual Information Technology, Electronics and Mobile Communication Conference (IEMCON)*. IEEE. 2018, pp. 1215–1220.

[134] C. Sturm, J. Szalanczi, S. Schönig, and S. Jablonski. "A lean architecture for blockchain based decentralized process execution". In: *International conference on business process management*. Springer. 2018, pp. 361–373.

[135] A. B. Tran, Q. Lu, and I. Weber. "Lorikeet: A Model-Driven Engineering Tool for Blockchain-Based Business Process Execution and Asset Management." In: *BPM (Dissertation/Demos/Industry)*. 2018, pp. 56–60.

[136] C. Di Ciccio, A. Cecconi, M. Dumas, L. García-Bañuelos, O. López-Pintado, Q. Lu, J. Mendling, A. Ponomarev, A. B. Tran, and I. Weber. "Blockchain support for collaborative business processes". In: *Informatik Spektrum* 42.3 (2019), pp. 182–190.

[137] C. Klinkmüller, I. Weber, A. Ponomarev, A. B. Tran, and W. van der Aalst. "Efficient logging for blockchain applications". In: *arXiv preprint arXiv:2001.10281* (2020).

[138] P. De Filippi, M. Mannan, and W. Reijers. "Blockchain as a confidence machine: The problem of trust & challenges of governance". In: *Technology in Society* 62 (2020), p. 101284.

[139] J. C. Recker. "X-aware business process management". In: *BP Trends* 8.12 (2011), pp. 1–7.

[140] M. Rosemann and J. C. Recker. "Context-aware process design: Exploring the extrinsic drivers for process flexibility". In: *The 18th International Conference on Advanced Information Systems Engineering. Proceedings of Workshops and Doctoral Consortium*. Namur University Press. 2006, pp. 149–158.

[141] M. T. Wynn, J. De Weerdt, A. H. ter Hofstede, W. M. van der Aalst, H. A. Reijers, M. J. Adams, C. Ouyang, M. Rosemann, and W. Z. Low. "Cost-aware business process management: a research agenda". In: (2013).

[142] A. Ghose, K. Hoesch-Klohe, L. Hinsche, L.-S. Le, et al. "Green business process management: A research agenda". In: *Australasian Journal of Information Systems* 16.2 (2010).

[143] B. H. Sheppard and D. M. Sherman. "The grammars of trust: A model and general implications". In: *Academy of management Review* 23.3 (1998), pp. 422–437.

[144] T. Grandison and M. Sloman. "Specifying and analysing trust for internet applications". In: *Towards the Knowledge Society*. Springer, 2003, pp. 145–157.

[145] S. Garfinkel. *PGP: pretty good privacy*. " O'Reilly Media, Inc.", 1995.

[146] *X.509: Information technology - Open Systems Interconnection - The Directory: Public-key and attribute certificate frameworks*. `https://www.itu.int/rec/T-REC-X.509`. (Accessed on 01/04/2021).

[147] M. Blaze, J. Feigenbaum, and J. Lacy. "Decentralized trust management". In: *Proceedings 1996 IEEE Symposium on Security and Privacy*. IEEE. 1996, pp. 164–173.

[148] R. Khare and A. Rifkin. "Weaving a web of trust." In: *World Wide Web Journal* 2.3 (1997), pp. 77–112.

[149] A. Josang, R. F. Hayward, and S. Pope. "Trust network analysis with subjective logic". In: (2006).

[150] A. Jøsang and T. Bhuiyan. "Optimal trust network analysis with subjective logic". In: *2008 Second International Conference on Emerging Security Information, Systems and Technologies*. IEEE. 2008, pp. 179–184.

[151] F. E. Walter, S. Battiston, and F. Schweitzer. "A model of a trust-based recommendation system on a social network". In: *Autonomous Agents and Multi-Agent Systems* 16.1 (2008), pp. 57–74.

[152] A. Xu and G. Dudek. "Optimo: Online probabilistic trust inference model for asymmetric human-robot collaborations". In: *2015 10th ACM/IEEE International Conference on Human-Robot Interaction (HRI)*. IEEE. 2015, pp. 221–228.

[153] F. Skopik, D. Schall, and S. Dustdar. "Start trusting strangers? bootstrapping and prediction of trust". In: *International conference on web information systems engineering*. Springer. 2009, pp. 275–289.

[154] Q. Zeng, S. X. Sun, H. Duan, C. Liu, and H. Wang. "Cross-organizational collaborative workflow mining from a multi-source log". In: *Decision support systems* 54.3 (2013), pp. 1280–1301.

[155] M. Müller, N. Ostern, and M. Rosemann. "Silver Bullet for All Trust Issues? Blockchain-Based Trust Patterns for Collaborative Business Processes". In: *Business Process Management: Blockchain and Robotic Process Automation Forum*. Cham: Springer International Publishing, 2020, pp. 3–18.

[156] ISO/IEC 27000. *International Standard ISO / IEC Information technology — Security techniques — Information security management systems — Overview and*. Tech. rep. ISO, 2018.

[157] R. Ross, M. McEvilley, and J. Oren. *Systems security engineering: Considerations for a multidisciplinary approach in the engineering of trustworthy secure systems*. Tech. rep. National Institute of Standards and Technology, 2016.

[158] Object Management Group. *UMLTM Profile for Modeling Quality of Service and Fault Tolerance Characteristics and Mechanisms Specification*. `https://www.omg.org/spec/QFTP/1.1/PDF`. (Accessed on 01/07/2021). 2008.

[159] N. G. Mohammadi, S. Paulus, M. Bishr, A. Metzger, H. Könnecke, S. Hartenstein, T. Weyer, and K. Pohl. "Trustworthiness attributes and metrics for engineering trusted internet-based software systems". In: *International Conference on Cloud Computing and Services Science*. Springer. 2013, pp. 19–35.

[160] M. Menzel, I. Thomas, and C. Meinel. "Security requirements specification in service-oriented business process management". In: *2009 International Conference on Availability, Reliability and Security*. IEEE. 2009, pp. 41–48.

[161] M. Backes, B. Pfitzmann, and M. Waidner. "Security in business process engineering". In: *International Conference on Business Process Management*. Springer. 2003, pp. 168–183.

[162] D. Xiang, G. Liu, C. Yan, and C. Jiang. "Detecting data-flow errors based on Petri nets with data operations". In: *IEEE/CAA Journal of Automatica Sinica* 5.1 (2017), pp. 251–260.

[163] E. Gamma. *Design patterns: elements of reusable object-oriented software*. Pearson Education India, 1995.

[164] R. C. Nickerson, U. Varshney, and J. Muntermann. "A method for taxonomy development and its application in information systems". In: *European Journal of Information Systems* 22.3 (2013), pp. 336–359.

[165] M. Markovska. *Modelling Business Processes on a Blockchain Eco- System (BPMN)*. Master Thesis. 2019.

[166] X. Liang, S. Shetty, D. Tosh, C. Kamhoua, K. Kwiat, and L. Njilla. "Provchain: A blockchain-based data provenance architecture in cloud environment with enhanced privacy and availability". In: *2017 17th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGRID)*. IEEE. 2017, pp. 468–477.

[167] B. Liu, X. L. Yu, S. Chen, X. Xu, and L. Zhu. "Blockchain based data integrity service framework for IoT data". In: *2017 IEEE International Conference on Web Services (ICWS)*. IEEE. 2017, pp. 468–475.

[168] P. K. Banerjee, P. Kulkarni, and H. S. Patil. *Distributed logging of application events in a blockchain*. US Patent 10,320,566. 2019.

[169] J. Hannemann and G. Kiczales. "Design pattern implementation in Java and AspectJ". In: *Proceedings of the 17th ACM SIGPLAN conference on Object-oriented programming, systems, languages, and applications*. 2002, pp. 161–173.

[170] V. Costan and S. Devadas. "Intel SGX Explained." In: *IACR Cryptol. ePrint Arch.* 2016.86 (2016), pp. 1–118.

[171] S. Pinto and N. Santos. "Demystifying arm trustzone: A comprehensive survey". In: *ACM Computing Surveys (CSUR)* 51.6 (2019), pp. 1–36.

[172] L. Zhang, S. Bakshi, and J. K. Zao. "Off-Chain Trusted Computing". In: *IEEE Internet of Things Magazine* 3.2 (2020), pp. 8–9.

[173] R. Cheng, F. Zhang, J. Kos, W. He, N. Hynes, N. Johnson, A. Juels, A. Miller, and D. Song. "Ekiden: A platform for confidentiality-preserving, trustworthy, and performant smart contracts". In: *2019 IEEE European Symposium on Security and Privacy (EuroS&P)*. IEEE. 2019, pp. 185–200.

[174] P. Resnick and R. Zeckhauser. "Trust among strangers in internet transactions: Empirical analysis of ebay's reputation system". In: *The Economics of the Internet and E-commerce* 11.2 (2002), pp. 23–25.

[175] E. Bellini, Y. Iraqi, and E. Damiani. "Blockchain-based distributed trust and reputation management systems: a survey". In: *IEEE Access* 8 (2020), pp. 21127–21151.

[176] M. Müller, S. R. Garzon, M. Westerkamp, and Z. A. Lux. "HIDALS: A Hybrid IoT-based Decentralized Application for Logistics and Supply Chain Management". In: *2019 IEEE 10th Annual Information Technology, Electronics and Mobile Communication Conference (IEMCON)*. IEEE. 2019, pp. 0802–0808.

[177] *Data Protection Act 2018*. `https://www.legislation.gov.uk/ukpga/2018/12/contents/enacted`. (Accessed on 02/26/2021).

[178] K. J. Blois. "Trust in business to business relationships: An evaluation of its status". In: *Journal of Management Studies* 36.2 (1999), pp. 197–215.

[179] P. De Filippi, M. Mannan, and W. Reijers. "Blockchain as a confidence machine: The problem of trust & challenges of governance". In: *Technology in Society* 62 (2020), p. 101284.

[180] A. P. Joshi, M. Han, and Y. Wang. "A survey on security and privacy issues of blockchain technology". In: *Mathematical Foundations of Computing* 1.2 (2018), pp. 121–147.

[181] P. Koshy, D. Koshy, and P. McDaniel. "An analysis of anonymity in bitcoin using p2p network traffic". In: *International Conference on Financial Cryptography and Data Security*. Springer. 2014, pp. 469–485.

[182] F. Victor. "Address clustering heuristics for Ethereum". In: *Financial Cryptography and Data Security*. Ed. by J. Bonneau and N. Heninger. Cham: Springer International Publishing, 2020.

[183] R. Mühlberger, S. Bachhofner, C. Di Ciccio, L. García-Bañuelos, and O. López-Pintado. "Extracting Event Logs for Process Mining from Data Stored on the Blockchain". In: *International Conference on Business Process Management*. Springer. 2019, pp. 690–703.

[184] F. Victor, P. Ruppel, and A. Kupper. "A Taxonomy for Distributed Ledger Analytics". In: *Computer* 54.2 (2021), pp. 30–38.

[185] I. Sommerville. "Software engineering 9th Edition". In: *ISBN-10* 137035152 (2011), p. 18.

[186] Camunda Services GmbH. *Camund - Workflow and Decision Automation Platform*. `https://camunda.com/`. (Accessed on 03/17/2021).

[187] P. J. Denning. "A new social contract for research". In: *Communications of the ACM* 40.2 (1997), pp. 132–134.

[188] S. Gregor and A. R. Hevner. "Positioning and presenting design science research for maximum impact". In: *MIS quarterly* (2013), pp. 337–355.

[189] P. Johannesson and E. Perjons. "Evaluate Artefact". In: *An Introduction to Design Science*. Cham: Springer International Publishing, 2014, pp. 137–149. isbn: 978-3-319-10632-8. doi: `10.1007/978-3-319-10632-8_9`. url: `https://doi.org/10.1007/978-3-319-10632-8_9`.

[190] L. Pan and C. Li. *6219 pairs of BPMN images and definition files*. 2020. doi: `10.21227/yzvb-0f30`. url: `https://dx.doi.org/10.21227/yzvb-0f30`.

[191] GenMyModel website. *Model Repository - Browse UML, BPMN, Database (RDS) and Flowchart examples*. `https://app.genmymodel.com/api/repository`. (Accessed on 12/01/2020).

[192] F. Corradini, F. Fornari, A. Polini, B. Re, and F. Tiezzi. "RePROSitory: a Repository Platform for Sharing Business PROcess modelS." In: *BPM (PhD/Demos)* 2420 (2019), pp. 149–153.

[193] E. Rolón, F. Ruiz, F. García, and M. Piattini. "Applying software metrics to evaluate business process models". In: *CLEI-Electronic Journal* 9.1 (2006).

[194] D. Adams. *The Hitchhiker's Guide to the Galaxy*. London: Baker, 1979.

[195] F. D. Davis. "Perceived usefulness, perceived ease of use, and user acceptance of information technology". In: *MIS quarterly* (1989), pp. 319–340.

[196] I. Azjen. "Understanding attitudes and predicting social behavior". In: *Englewood Cliffs* (1980).

[197] C. Gorenflo, S. Lee, L. Golab, and S. Keshav. "FastFabric: Scaling Hyperledger Fabric to 20,000 Transactions per Second". In: *2019 IEEE International Conference on Blockchain and Cryptocurrency (ICBC)*. 2019, pp. 455–463. doi: 10.1109/BLOC.2019.8751452.

[198] M. Abdelhamid, J. Gaia, and G. L. Sanders. "Putting the focus back on the patient: how privacy concerns affect personal health information sharing intentions". In: *Journal of medical Internet research* 19.9 (2017), e169.

[199] R. H. Chipika, E. Finegan, S. Li Hi Shing, O. Hardiman, and P. Bede. "Tracking a fast-moving disease: longitudinal markers, monitoring, and clinical trial endpoints in ALS". In: *Frontiers in neurology* 10 (2019), p. 229.

[200] *iExec Blockchain-Based Decentralized Cloud Computing*. https://iex.ec/. (Accessed on 06/03/2021).

[201] J. Benet. "Ipfs-content addressed, versioned, p2p file system". In: *arXiv preprint arXiv:1407.3561* (2014).

[202] J. Freund and B. Rücker. *Praxishandbuch BPMN: mit Einführung in CMMN und DMN*. Carl Hanser Verlag GmbH Co KG, 2016.