TU WIEN Informatics

Time-travelling State Machines for Verifiable BPM

DIPLOMARBEIT

zur Erlangung des akademischen Grades

Diplom-Ingenieur

im Rahmen des Studiums

Software Engineering und Internet Computing

eingereicht von

Daniel Kleebinder, BSc
Matrikelnummer 51832684

an der Fakultät für Informatik

der Technischen Universität Wien

Betreuung: Univ.Prof. Dipl.-Ing. Dr.techn. Wolfgang Kastner
Mitwirkung: Dipl.-Ing. Thomas Preindl, BSc
Dipl.-Ing. Martin Kjäer, BSc

Wien, 1. Dezember 2022

Daniel Kleebinder   Wolfgang Kastner

Technische Universität Wien
A-1040 Wien ▪ Karlsplatz 13 ▪ Tel. +43-1-58801-0 ▪ www.tuwien.at

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

Informatics

Time-travelling State Machines for Verifiable BPM

DIPLOMA THESIS

submitted in partial fulfillment of the requirements for the degree of

Diplom-Ingenieur

in

Software Engineering and Internet Computing

by

Daniel Kleebinder, BSc
Registration Number 51832684

to the Faculty of Informatics

at the TU Wien

Advisor: Univ.Prof. Dipl.-Ing. Dr.techn. Wolfgang Kastner
Assistance: Dipl.-Ing. Thomas Preindl, BSc
Dipl.-Ing. Martin Kjäer, BSc

Vienna, 1st December, 2022

Daniel Kleebinder   Wolfgang Kastner

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

Erklärung zur Verfassung der Arbeit

Daniel Kleebinder, BSc

Hiermit erkläre ich, dass ich diese Arbeit selbständig verfasst habe, dass ich die verwendeten Quellen und Hilfsmittel vollständig angegeben habe und dass ich die Stellen der Arbeit – einschließlich Tabellen, Karten und Abbildungen –, die anderen Werken oder dem Internet im Wortlaut oder dem Sinn nach entnommen sind, auf jeden Fall unter Angabe der Quelle als Entlehnung kenntlich gemacht habe.

Wien, 1. Dezember 2022

Daniel Kleebinder

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

Danksagung

Ich möchte meinem Betreuer Wolfgang Kastner für all seine wertvollen Ratschläge und die Möglichkeit, Teil der Automation Systems Group (ASG) zu werden, danken. Die gemeinsame Arbeit an einigen Forschungsprojekten und das Miterleben echter Wissenschaft war eine Erfahrung, für die ich sehr dankbar bin. Außerdem möchte ich Thomas Preindl und Martin Kjäer für ihre Ausdauer, ihr kontinuierliches Feedback und vor allem unsere spannenden Diskussionen danken. Aufgrund ihrer unermüdlichen Unterstützung wurde diese Arbeit zu dem, was sie heute ist.

Darüber hinaus möchte ich mich bei allen anderen an der Universität Tätigen dafür bedanken, dass ich die Chance erhalten habe, diese faszinierende Ausbildung genießen zu dürfen. Das vielfältige Curriculum, das häufig mit Begeisterung und Freude gelehrt wurde, führte so auch zu unvergesslichen Momenten, an die ich mich gerne zurückerinnere.

Abschließend möchte ich meinen Eltern Franz und Margit dafür danken, dass sie mir während meines Studiums immer mit Rat und Tat zur Seite gestanden sind, meinen Brüdern, mit denen ich auch in schwierigen Zeiten noch etwas zu lachen gefunden habe, und meiner Freundin Cornelia für ihre Ermutigungen und Unterstützung. Danke, dass ihr für mich da seid und mir geholfen habt, in dieser Arbeit wie auch in meinem Studium bis zum Schluss mein Bestes zu geben.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

Acknowledgements

I want to thank my advisor Wolfgang Kastner for all his valuable input and the chance to become part of the Automation Systems Group (ASG). The joint work on some of the research projects and witnessing real scientific labor was an experience I am very grateful for. Furthermore, I want to thank Thomas Preindl and Martin Kjäer for their perseverance, constant feedback, and especially our exciting discussions. Due to their relentless work and support, this thesis has become what it is today.

Moreover, I want to express my gratitude to everyone else being part of the university for allowing me to attend this fascinating education and for developing a manifold curriculum that was taught with enthusiasm, often leading to memorable moments.

Finally, I want to thank my parents, Franz and Margit, for their guidance, emotional and financial support; my brothers, who were always up for laughter even when times got tough; and my girlfriend, Cornelia, for her advice and encouragement. Thank you all for being there for me and helping me finish this thesis and my studies.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

Kurzfassung

Bei der Verwaltung von Geschäftsprozessen sind Unternehmen häufig auf Dritte angewiesen, um Vertrauen zwischen allen Beteiligten schaffen zu können. Seit dem Aufkommen der Blockchain-Technologie zielen Forschende weltweit darauf ab, Blockchains als eine eben solche vertrauenswürdige dritte Partei einzusetzen, um auf deren strikte Nachvollziehbarkeit bauen zu können. Diese Ansätze sind allerdings meist sehr stark mit der Blockchain selbst verzahnt, was zu mangelnder Flexibilität oder erhöhten Kosten führen kann. Aus diesem Grund wird in dieser Arbeit ein neuartiges Konzept vorgestellt, das die Ausführung und Überprüfung von Geschäftsprozessen mit Hilfe der Blockchain zwar ermöglicht, Eigenschaften wie Flexibilität aber dennoch erfüllt.

Das Konzept selbst basiert auf einer modularen Softwarearchitektur, bei der jedes Modul über ein Event-Sourcing-System lose mit anderen Modulen gekoppelt ist. Es zielt darauf ab, die Vorteile einer Blockchain zu bewahren, indem es Beteiligten unter anderem erlaubt, vergangene Zustände zu verifizieren, wobei der Beweis für die Korrektheit dieser Zustände allerdings auf der Blockchain selbst zu finden ist. Dieser Ansatz wurde mithilfe bestehender Literatur entwickelt und später als Prototyp umgesetzt.

Anschließend wurde das Konzept und der entsprechende Prototyp anhand qualitativer Kriterien bewertet, und die Komplexität durch Anwendung formaler Methoden auf deren statische Strukturen analysiert. Nachfolgend wurde der praktische Nutzen mithilfe von vereinfachten Geschäftsprozessen aufgezeigt. Im Vergleich zu bestehenden Ansätzen ergaben sich drastische Verbesserungen in Bezug auf Flexibilität und Datenschutz. Weiters beträgt die mittlere Dauer einer Geschäftsprozesstransaktion auf Ethereum etwa 18 Sekunden, was diesen Ansatz für, sowohl lang- als auch kurzlebige, Geschäftsprozesse nutzbar macht. Dennoch bleiben auch Nachteile bestehen. Die Geschäftsprozesstransaktionskosten korrelieren linear mit der Anzahl der Teilnehmenden, was bei kleineren Geschäftsprozessen zwar zu einer Gesamtkostenreduktion führen kann, bei größeren allerdings eine Teuerung darstellt.

Die Idee hinter dieser Arbeit ist nicht nur die Vorstellung eines neuen Konzepts zur Ausführung und Verifizierung von Geschäftsprozessen mithilfe der Blockchain, sie soll auch zukünftige Arbeiten dazu anregen, die Blockchain selbst nur noch als lose gekoppeltes Subsystem zu betrachten, um den Nachteilen solcher Systeme begegnen zu können.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

Abstract

Inter-organizational business process management often relies on third parties to establish trust between participants. Since the rise of blockchain technology and its associated properties regarding traceability, research communities aim to integrate blockchains into workflow execution engines in favor of a trusted third party. Frequently, these approaches directly leverage upon the blockchain, which leads to shortcomings such as a lack of flexibility or increased cost. Therefore, this thesis proposes a novel concept that allows workflow execution and verification using the blockchain while preserving flexibility and reducing transaction costs by utilizing present-day cryptography.

The concept relies on a modular software architecture where each module is loosely coupled to others through an event-sourcing system. It aims to preserve the advantages of a blockchain by enabling time travel to allow participants to verify past states while keeping proof of the correctness of these states on the blockchain, which acts as a single source of truth. The proposed approach was derived by discussing related literature and exploring its practical utility by creating a prototypical implementation.

The proposed concept and its prototype were evaluated against qualitative criteria extracted from related work, while the complexity was analyzed by applying formal methods to static structures. Afterwards, their practical utility was exhibited by executing simulations of simplified real-world business processes. The evaluation has shown dramatic improvements regarding flexibility and privacy. Furthermore, the execution duration per transaction was observed to be around 18 seconds on Ethereum. This qualifies the proposed concept as a suitable approach for slow- and fast-paced business processes. Nonetheless, some shortcomings remain. The execution cost linearly correlates to the number of participants, which results in an overall cost reduction for smaller business processes but breaks even and exceeds other approaches after reaching a certain threshold.

This work presents a novel idea for workflow execution that leverages on properties of blockchains and instigates future work to demystify and treat blockchains only as loosely coupled sub-systems of supportive nature.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

Contents

Kurzfassung **xi**

Abstract **xiii**

Contents **xv**

1 Introduction **1**
1.1 Aim of the Work 2
1.2 Methodological Approach 3
1.3 Structure of the Work 4

2 Background **5**
2.1 Consensus 5
2.2 Blockchain-oriented Software Engineering 11
2.3 Onchain vs Offchain 15
2.4 Business Process Management 19
2.5 Baseline Protocol 23

3 Related Work **27**
3.1 Literature Review Methodology 27
3.2 Concept Comparison 39

4 Time-travelling State Machines **43**
4.1 Design Science Methodology 43
4.2 Proposed Concept 45
4.3 Prototype Design 60
4.4 Intrinsic Properties 73

5 Evaluation **77**
5.1 Qualitative Analysis 77
5.2 Static Analysis 84
5.3 Scenario Simulations 88
5.4 Integration with Camunda's Zeebe 102

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

6 Conclusion 107

List of Figures 109

List of Tables 111

List of Algorithms 113

List of Listings 115

Acronyms 117

Bibliography 119

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

CHAPTER 1

Introduction

Recent years have shown an ever-growing interest in Business Process Management (BPM) from both industrial and research communities. Members of these can be categorized into sub-groups with very different educational backgrounds, each applying BPM in their own way and coming up with their own very specific requirements. Business administrations, for example, aim to use methodical BPM to optimize workflows to reduce cost and increase customer satisfaction. Software communities that must implement such workflows using existing technologies strive for robust but flexible solutions to keep up with changing requirements introduced by business administrations or by legal authorities [Wes12b]. To bridge the gap between these two worlds, computer science research communities devised notations and formal specifications that create a common ground [OMG11, Wes12c, Wes12d, LFW20].

Nonetheless, some open problems still need to be solved, especially regarding inter-organizational BPM due to a lack of mutual trust between counterparties. It is not uncommon that cooperating participants are in a conflict of interest that hinders the progression toward a set goal. The impact of trust missing as a key ingredient in collaborative business processes like supply chains [JMSK04], health care [KJ21] or logistics [FVB05] has been studied thoroughly where all research came to the same conclusion of trust being a factor that cannot be underestimated; neither in social nor inter-organizational relationships [PL09]. Recent years, however, have shown an innovation emerge that has the potential to revolutionize how trust is handled — introducing Blockchain Technology (BCT).

Roughly outlined as a tamper-proof series of timestamped transactions and famously known for cryptocurrencies such as Bitcoin [Nak09] and Ethereum [But22], blockchains are also a viable tool in the arsenal of BPM. A lot of potential use cases have been discussed and demonstrated, ranging from the use in workflow execution systems to monitoring of business processes, from governmental environments to the Internet of Things (IoT) [MWA+18, WXR+16, SSSJ19, LBAG21, VXBP19]. However, almost all

TU Bibliothek Your knowledge hub WIEN
Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

of these approaches leverage the blockchain as a first-level citizen and directly build on top of it using smart contracts which inevitably leads to privacy concerns and drastically increased cost [Woo22]. On blockchains, the execution of workflow steps, that only a subset of participants is concerned with, must nonetheless be executed by all participants in the same way. This exposes potentially confidential information and increases execution cost. Keeping internal processes private, however, is of utmost importance for many companies and organizations [FSM21, CRF18].

Furthermore, concepts tightly coupled to the blockchain, in the form of smart contracts, for example, come with a rather significant flexibility penalty. Changing workflow participants, the workflow definition, or adapting capabilities of the underlying workflow execution system is a complex and costly task [Woo22, SAW20]. This means that business processes that require a high level of flexibility can only take advantage of blockchain-intrinsic properties, such as trust decentralization or transparency to some extent which reduces acceptance of these approaches.

1.1 Aim of the Work

To tackle the aforementioned open problems in inter-organizational BPM and the lack of standardized workflow system architectures [Pry16], this work proposes a novel concept for a "time-travelling"[1] state machine that enables execution of business processes including multiple participants in an environment that provides unconditional mutual trust. The concept aims to provide a modular software architecture that allows the integration of different BCTs to leverage upon their diverse advantages and to enable integration of future blockchains or layer-2 rollups to prevent participants from being locked to a particular blockchain. Additionally, and in contrast to most existing approaches, the concept provides a high level of flexibility regarding system architecture, workflow participant selection, and workflow definition. To achieve this goal, the following research questions are answered throughout the course of this work.

RQ1 What is the state of the art for BCT-based state machines for business process engines?

RQ2 Which properties do BCT-based state machines require to allow time-travel verification of business processes?

RQ3 Which aspects must be adapted to close the gap between the state of the art and a privacy-preserving BCT-based state machine that allows time-travel verification?

The first research question is answered in section 3.2.1 after conducting a thorough literature review in chapter 3. It should give the reader an overview of existing approaches and identify the gaps in the state of the art. Building upon research question 1, the

[1] Being able to jump between active and past states to perform business process validation.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Wien Bibliothek Your knowledge hub

second research question is answered in section 4.4.1 by identifying unique properties of the proposed concept derived from the state of the art and requirements identified in real-world industrial use cases. The last research question aims to answer how well the proposed concept can be integrated into existing solutions, a property much expected by users of such a system, and which aspects to adapt in order to make it work.

1.2 Methodological Approach

The following methodologies are employed to answer the research questions, ensure scientific rigor and allow for reproducibility of the results. Tailoring has been applied and described in detail at the beginning of the related work chapter in section 3.1 and at the beginning of the chapter that describes the proposed concept in section 4.1.

1.2.1 Literature Review

A narrative literature review is performed to accumulate background knowledge of topics including BPM, BCTs, and blockchain-oriented software engineering. Furthermore, work related to workflow execution on the blockchain is investigated and compared with each other in more detail. A predefined set of search words and literature databases such as IEEE and ResearchGate are used to improve reproducibility of the results [RA11, Str19]. The narrative literature review is used to answer RQ1 and lays the foundation for RQ2 and RQ3.

1.2.2 Design Science

The remainder of this work follows the design science research approach for information systems [HMPR04]. A concept for a time-travelling state machine is derived in an iterative process. The concept is implemented and evaluated against the state of the art by (1) determining common metrics from related literature in a qualitative analysis process, (2) performing a static analysis of the software architecture to derive its complexity, and (3) simulating real-world use cases in experiments to determine the utility of the produced artifacts. The artifacts produced are (1) a concept for a flexible and robust time-travelling state machine and (2) an instantiation of this concept in the form of a prototype that future work can extend upon and integrate into existing solutions. The research results are communicated to management- and technology-oriented audiences by publishing this work. Design science and the produced artifacts are used to answer RQ2 and RQ3.

1.2.3 Software Engineering

Software engineering methodologies are employed during the design, implementation, and evaluation phases. Requirements engineering is used in earlier stages of this work to create and determine industrial real-world use cases that the concept is designed and later evaluated against. BPMN and UML are used throughout the course of the entire

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

work to visualize business processes and software architectures. Other methodologies such as event sourcing, unit- and integration testing, object-oriented, functional and reactive programming are used to implement the prototype. Following standard and state-of-the-art software engineering approaches allow industry and research developers to properly implement a time-travelling state machine, leveraging their existing knowledge base. These methodologies help to answer RQ2 and RQ3.

1.3 Structure of the Work

The remainder of this work is structured in a way that introduces the reader to the topics of BPM and BCTs, to then transition into a concept for a time-travelling state machine followed by its evaluation. Background knowledge in chapter 2 gives a brief introduction into BCTs and consensus algorithms in section 2.1 followed by a description of software engineering approaches that are applicable for the domain of BCTs. The remainder of this chapter focuses on notations commonly used in BPM in both industrial and research settings.

Thereafter, the reader is introduced to approaches from literature related to this work's contribution in chapter 3. At the beginning of this chapter, the narrative literature methodology and its tailoring are introduced, followed by detailed descriptions of some of the more relevant approaches found. Each approach is briefly introduced and explained, followed by a short discussion of advantages and disadvantages. The chapter concludes by comparing the presented approaches with the proposed concept of this work in section 3.2 and answers RQ1 in subsection 3.2.1.

Chapter 4 then introduces the tailoring applied to the design science methodology in section 4.1, followed by a thorough description of a time-travelling state machine in section 4.2. This is the first artifact produced in this work. The second artifact, the instantiation, is described in the prototype design in section 4.3. It introduces all used technologies, followed by a per-module description of the implementation. This chapter concludes by listing intrinsic properties of a time-travelling state machine as observed in the proposed concept and the prototype design in section 4.4 and afterwards answers RQ2 in subsection 4.4.1.

The evaluation in chapter 5 begins with a qualitative analysis where common metrics are derived from related literature and are applied to the proposed concept. This is followed by a static analysis in section 5.2 that focuses on formal metrics and the analysis of the software architecture. Afterwards, real-world scenarios are simulated to demonstrate utility in section 5.3, and the integration into existing systems in section 5.4. Each section summarizes and discusses its results at the end. Chapter 5 concludes by answering RQ3.

The last chapter 6 concludes this work by briefly outlining the proposed concept and summarizing the evaluation results. Furthermore, it states problems that remain unsolved, follow-up research questions that came up during the course of this work, and opportunities for future work are also discussed.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

CHAPTER 2

Background

The background chapter introduces fundamental concepts that are required throughout the rest of the work. This includes BPM, BCTs, consensus algorithms, notations and blockchain-oriented software engineering approaches.

2.1 Consensus

The upcoming section will give a small glimpse behind the concept of blockchains, it will introduce some consensus techniques for distributed computer systems, and explain the difference between specific leader-election mechanisms for blockchains such as Proof of Work (PoW) and Proof of Stake (PoS).

2.1.1 Blockchain

A blockchain is a linked list of records (so-called “blocks”), where each record itself contains a list of transactions and the hash of the previous record originating from the so-called “genesis block”. The links between records introduce unique properties such as immutability, transparency, and produces a traceable list that others can verify using cryptographic procedures. This creates an append-only data store that is typically distributed over a peer-to-peer network where each peer keeps a (full) copy of the history of transactions dispatched to the blockchain. In distributed environments, new blocks are typically determined and attached to the blockchain using consensus protocols that prevent Byzantine faults and establish trust. If blockchains are used in a financial context, they are sometimes referred to as Distributed Ledger Technology (DLT) as well [SLHK19, WZ18]. Satoshi Nakamoto was the first to utilize them in a decentralized, trusted finance system in the form of Bitcoin. The core concept of Bitcoin is to provide a solution that solves the double-spending problem of virtual currencies and does not rely on a centralized, trusted third party [Nak09].

2.1.2 Proof of Work (PoW)

Bitcoin relies on PoW to elect a trusted proposer. Proposers are allowed to create new blocks (i.e., a list of transactions and some metadata) based on the entirety of collected transactions so far. This block will then be distributed to the network using a form of total order broadcasting, a technique that ensures, that all participants receive the same blocks in the exact same order, and will be verified by other nodes that prevent double-spending attacks in case the proposer wants to abuse the fact of being trusted by others.

PoW is part of the Nakamoto consensus introduced in the Bitcoin whitepaper and is one of the many consensus techniques available. Miners compete on who is allowed to create the next block in the chain to earn (1) the right for coinbase transactions (the first transaction in a block that creates new units of said currency) and (2) the transaction fees. The more mining power a miner can utilize, the more often said miner will be elected as block proposer. Typically, Bitcoin aims to keep a block time of around 10 minutes (i.e., it should take the entire pool of miners 10 minutes to mine a new block). This is achieved by dynamically adjusting the amount of work that has to be done depending on the network's total computing power. Therefore, the more computing power available, the harder the problem will become to solve (and vice versa in case miners leave the network). To be more formal, Nakamoto consensus (similar to Ethereum's consensus technique) wants miners to find a nonce n such that

$$H(n||H(b)) < l \tag{2.1}$$

where l specifies the upper bound for the hash value produced by the hash of the previous block b concatenated with the nonce. Once such an n is found, other miners can easily validate the result. The algorithm adjusts the difficulty by adjusting how small l is [SLHK19]. This technique, however, became a target of certain controversy. The strong incentive of earning Bitcoins from mining new blocks leads to an uprise of miners in the network. The more miners, the harder the problem will become to solve. This increases the networks total energy consumption proportionally to the computational power available, causing the entirety of the Bitcoin ecosystem to consume approximately 99.27 TWh in the year 2022. To put this in perspective, the country of Austria will consume around 64.61 TWh in the same year according to predictions [Cam22].

Another controversy regarding PoW in the context of Bitcoin is an attack vector called "Selfish Mining" that allows miners to earn disproportionately more revenue compared to their provided computational power. The idea behind this attack is to assume that miners will strategically collude in order to maximize the computing power to Bitcoins earned ratio. This is achieved by hiding successfully mined blocks from the main net, publishing them at certain times in the future, and thus wasting the computational power of trustful miners because the main net will always switch to the subchain that requires the most computational power (often referred to as the longest chain) [ES13].

TU Bibliothek Your knowledge hub WIEN
Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

2.1.3 Proof of Stake (PoS)

PoS is one of the most popular alternatives to PoW. Even though it is similarly used as a leader-election mechanism, it does this in a much more energy-efficient way and thus overcomes one of the shortcomings of PoW. The idea behind this concept is that the more a node has at stake (be it in the form of an arbitrary resource), the more invested said one is to keep the system up and running correctly. This is achieved by choosing the next proposer based on the number of coins and the age of each coin. Nodes with a rather active wallet that contribute much to the network will thus be selected more often to propose a new block. If some node acts maliciously, the chances of being selected are reduced by the algorithm itself [SLHK19].

Even PoS comes with its quite unique downsides. One of those is the Nothing-at-Stake problem. It states that the validators (the equivalent in PoS to miners in PoW) of the network have an intrinsic financial incentive to participate in every fork of the network because doing so has no downsides but only increases the amount of reward a validator can collect. In practice, however, it is assumed that validators are aware that intentional network forking would raise doubts about the system itself, decreasing the coin's value. Each fork would live with a separate ledger containing different values for each participant, thus creating artificial inflation that further reduces the trust in the network. A reduced coin value will inevitably hurt the holders. Since (in PoS) the set of coin holders is exactly the set of participating nodes, it is intrinsic to the participant to prevent any value reduction and therefore does not perform Nothing-at-Stake attacks. Nonetheless, the Ethereum network introduced "slashing", a mechanism to destroy a certain percentage of the hostile participants stake, to further reduce the risk for such attacks [Sal20].

2.1.4 State Machine Replication (SMR)

State Machine Replication (SMR) is a concept at the heart of any distributed, fault-tolerant consensus system. First introduced by Leslie Lamport in [Lam78], given a set of rules, SMR ensures that each replica is in the same state after a message was sent from a client to perform some action. Those rules are as follows:

- Each non-faulty replica starts in state s_0
- Each non-faulty replica that applies operation o to state s will end up in state s'
- Each operation of each non-faulty client is executed
- Each non-faulty replica executes the same order of operations $o_0, \ldots, o_i$

Each replica fulfills the first two properties without the need for any other distributed protocol. The latter two require some sort of total order broadcasting. Given that total order broadcasting can be reduced to the problem of consensus, some sort of distributed consensus protocol is required to establish a common order of operations [SB12].

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

2.1.5 Consensus Protocols

A problem that emerged with distributed computer environments is achieving consistency in synchronous and asynchronous settings assuming that one or many nodes are faulty. This is known as the consensus problem of computer science.

Lamport et al. [LSP02] exemplified this problem by using the metaphor of Byzantine generals[1] who have to decide on whether to attack or retreat in a battle. These generals, however, can only communicate with each other indirectly, causing delays and the possibility of messages getting lost. The authors have shown that at most f generals are allowed to be traitors when there is a total of $n = 3f + 1$ generals on the field. If there are more than f traitors, the consensus problem becomes undecidable. In computer networks, computing nodes, or processes, represent the generals. A node that, either intentionally or unintentionally, sends wrong messages to the other nodes or is holding them back is therefore Byzantine faulty.

However, one has to distinguish between Byzantine fault tolerance and crash tolerance. Byzantine faulty nodes typically cause more damage to the system than nodes that simply crash. To make a system crash tolerant, it is sufficient if the total number of nodes is $n = 2f + 1$. If f nodes crash, the remaining $f + 1$ nodes outnumber the crashed ones, and the system can still come to a consensus (assuming that Byzantine faulty nodes are no possibility in the given environment). Consensus protocols at least have to satisfy the following properties to be Byzantine or crash tolerant [SB12]:

- **Agreement**. Each non-faulty node must agree on the same value. They must not diverge; otherwise, the node is considered faulty.
- **Integrity**. Each non-faulty node can only decide once (i.e., once a value was decided, it is finalized and cannot be changed).
- **Termination**. Each non-faulty node eventually makes a decision (i.e., clients will get a response without the system getting stuck in an endless loop).

Additionally, to the properties above, some literature also adds the **Validity** property. Validity ensures that the value a node decides on was proposed by some other node. This rules out the trivial possibility that the deciding node always returns 0 (or some other predetermined constant) for any arbitrary input given.

Agreement, integrity, and validity are typical safety properties of a system, whereas termination is a liveness property [Kle17].

[1] Lamports initial intention was to name this problem “The Albanian Generals Problem”. In order to offend no readers, he was advised to use some no longer existing nationality like Byzantium, for example [Kle17].

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

2.1.6 Synchronous vs. Asynchronous System Models

Consensus literature distinguishes between synchronous and asynchronous system models and puts their work in either of both spotlights (or in both at the same time) to guarantee certain properties such as termination, for example.

The main difference between both is that synchronous systems proceed in a step-by-step manner. Those steps can be compared to moves in a chess game. Regardless of the time it took for each player to make a move, after each move, the chessboard is in a new and consistent state, and all participating parties (e.g., the referee, the players, and the audience) are aware of said new state. In distributed computer systems, such steps are often modeled with time frames where timeouts enable the system to proceed to the next frame. Compared to synchronous systems, asynchronous systems do not need to satisfy timeliness properties. Algorithms modeled with asynchronous systems in mind are typically more general since they do not rely on the performance of the host system they are operated on. Since no communication is instantaneous, asynchronous system models typically reassemble the real world quite well. System models that have both synchronous and asynchronous properties are typically referred to as semi-synchronous or semi-asynchronous systems [Agu10, CL99, Kle17].

2.1.7 Public vs. Private

From a philosophical point of view, consensus protocols and DLTs can be classified into three distinct categories. Namely public, private, and consortium networks. Each of these has unique properties applicable to certain scenarios. Public DLTs, for example, are usually favored in use cases where full decentralization and data integrity are desirable. The open consensus process allows anyone to validate the state of the DLT at any given point in time and report erroneous states if necessary [SLHK19]. Even though public DLTs are highly decentralized in theory because participation is open to anybody, in practice, some degree of centralization may occur in the form of colluding miner pools nonetheless. This is something public DLTs (such as Bitcoin and Ethereum) must consider to keep their system tamper-proof [Nak09, But22, ES13].

On the other hand, private DLTs are the complete opposite of public ones. Only authorized nodes are allowed to participate in the consensus process of the system. Some predetermined instances of a given organization will grant this permission to new participants. Even though read-only access could be granted to the public, private DLTs typically choose not to, which gives a better sense of privacy. On the other hand, writing is entirely restricted to a pre-selected set of participants. This allows for finality because the number of participating nodes is well known but also makes the system prone to tamper attempts because trust is required in the organization that selects the participants [SLHK19].

Consortium DLTs are DLTs where multiple organizations (that might be in conflict of interest) come together to form a network that is partially decentralized and otherwise shares most of the properties of private ones. Which kind of DLT and consensus

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

protocol to choose from highly depends on the use case and who are the participants and organizations involved [SLHK19].

2.1.8 Permissioned vs. Permissionless

In consensus, the terms permissioned and permissionless refer to whether participation in a consensus protocol is restricted or unrestricted. Permissioned consensus requires a central authority (or a group of authorized participants) that can (1) authenticate each other as members of the group and (2) add new members. Permissioned consensus protocols that are publicly available can be a target for Sybil attacks[2] and are therefore often replaced with permissionless consensus protocols (especially in the context of Blockchain technology). In contrast to permissioned consensus, permissionless consensus does not require any central authority. Bitcoin and Ethereum, for example, establish permissionless consensus through PoW and PoS. Sybil attacks are therefore ruled out by no longer relying on "who you are" but on "what you have" [Nak09, Sal20]. Permissionless consensus is often favored in public networks, while permissioned consensus is favored in private networks allowing a subset of members to strictly control who is allowed to participate [SLHK19].

2.1.9 FLP Impossibility Result

Named after the authors (Fischer, Lynch, and Paterson), the FLP impossibility result shows that consensus becomes undecidable in a purely deterministic and asynchronous system if at least one node fails. Fischer et al. have been awarded with the Dijkstra prize for the significance of this result [FLP85].

According to the FLP theorem, all asynchronous consensus protocols have to decide on which two of the following three properties to guarantee [SLHK19]

- **Safety**. When given the same input, all nodes should produce the same output (thus satisfying agreement, integrity, and validity).
- **Liveness**. A client will eventually receive a response to its request (and thus satisfying the termination).
- **Fault tolerance**. Even though not a must for consensus protocols, fault tolerance is still a property that is very much desirable. It allows a system to tolerate up to a certain number of faults f [CL99, BKM18, BSA14].

In highly distributed environments (especially in Blockchain technologies), implementations cannot forfeit fault tolerance at any cost. The threat of nodes being faulty (or even malicious in the sense of Byzantine faults) leads the implementation to choose between the safety and the liveness property. Practical Byzantine Fault Tolerance (PBFT), for

[2] One entity pretends to be many.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

example, strongly relies on the asynchronicity of safety and fault tolerance. In this case, the synchronicity of liveness is achieved by using timeouts and view changes and thus applying a step-by-step iteration as required in synchronous system models [CL99].

2.1.10 Tendermint

Recent years led to a boom of new Byzantine fault tolerant system proposals in academia that suggest PoS as the "better-suited" alternative to PoW to overcome some of the mentioned shortcomings. The Tendermint consensus protocol (based on PBFT [CL99]), for example, suggests a purely deterministic and predictable algorithm based on who owns how much money to decide who is allowed to propose the next block. This kind of gossip protocol based on PoS reduces the overall electricity consumption and the system's complexity.

Similar to Bitcoin and Ethereum, Tendermint also relies on the strict assumption that less than $\frac{1}{3}$ of all nodes are Byzantine faulty (i.e., produce wrong results or none at all either by accident or on purpose). Given that $n \geq 3f + 1$ is the minimum number of nodes, not more than f nodes are allowed to be faulty. Otherwise, the Tendermint protocol cannot guarantee safety and liveliness properties for consensus [BKM18].

2.2 Blockchain-oriented Software Engineering

With the growing interest in DLTs, which allow users to run customized code in a distributed and trustless environment in the form of Executable Distributed Code Contracts (EDCCs) as Ethereum does with smart contracts, systematic software development must be expanded upon in the context of blockchain technology [WEMG19]. Security, reliability, software architecture, and privacy are central aspects that must be considered when building applications using DLTs [PPMT17].

Methodical software engineering is an applicable approach to these kinds of challenges. It is the engineering discipline that is concerned with all stages of building software products and the application of systematic engineering approaches to software development itself. Software engineering can be split into four distinct activities that are sometimes also referred to as the software process [Som10, BF14]:

- **Specification**. Together with the customers and users, software requirement engineers define constraints, functional and non-functional requirements that the resulting software product has to fulfill in order to achieve some business value.

- **Development**. Software architects design the technical outline of the product given a certain specification, and developers implement this architecture using a set of requirements.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

- **Validation**. Software testers check if the product fulfills all specified requirements and therefore the customers and users needs. If this is not the case, the product is again modified according to the requirements.
- **Evolution**. The finished product is further iterated upon to reflect the changing market and customer requirements.

The exact details of each software process step and the process itself depend on the specific use cases and the domain in which the product is being developed in (e.g., the software process of an e-commerce and a spacecraft control system are most likely going to diverge) [Som10].

When building software applications in the context of blockchain technologies, these process steps must be extended upon to deal with newly emerging challenges. Organizations focusing on blockchain-oriented software development will feel the urge to introduce new professional roles that fill the gap between business and financial aspects and technological expertise. Due to the strong focus on financials, security is also of utmost importance [PPMT17]. EDCCs that malfunction may lead (and already have led) to tremendous losses. Hence, new formal methods must be devised that can be used to provide a profound tooling suite that is capable of analyzing, testing, and detecting programming errors in EDCCs. Especially the open source community has already developed some quite promising tools that are concerned with such issues. These tools are capable of performing static or dynamic byte or source code analysis to detect common security issues (examples of such are front-running, random numbers, or timestamp dependence), exploits, or formal guarantees. However, even more, and better tools that build upon existing open source technologies are expected to be developed in the future [dAS19].

Creating specifications, documentation, or test plans for blockchain-oriented applications might require some modeling language to better visualize and understand certain problems. Therefore, Blockchain-oriented Software Engineering (BoSE) suggests to expand existing models such as UML into the domain of DLTs. The newly created modeling language can then be further used to specify the software architecture. Besides defining a design notion, software architects will also be concerned with choosing which blockchain to use and how much blockchain truly is required for a certain use case (e.g., it might be sufficient if only the hash values of some files are stored instead of storing the content of all files). Defining new macro-architecture design patterns for the definition and development process will also be of utmost importance. Some already existing patterns and design approaches are discussed in the sections below [PPMT17, WEHG18].

2.2.1 Hybrid App Design

Complex software applications typically consist of multiple loosely coupled components that each strive for high cohesion. A more informal definition for this circumstance would be "what changes together, stays together" [New19]. These properties allow each component to be reused through standardized APIs and maintained without affecting

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

other components, therefore keeping change concentrated in a single spot. Following the low coupling high cohesion principle, new software engineering approaches, in particular *Chaos Engineering*[3] [RJ20], emerged in the industry that have been quickly adapted by others. Other principles such as *Acyclic Dependencies* and *Reuse Release Equivalence* are also rather common when talking about software application decomposition and domain-driven design in modern software architecture [Mar17].

Building upon mentioned principles, a new approach was derived specifically for creating hybrid applications where only certain components must communicate with the blockchain directly. To achieve this goal, Florian Blum et al. [WEHG18] suggest a three-step process to identify specific properties that are later used to derive the system architecture:

1. **Identify participants**. Defines the boundaries for a specific use case (and in parts for the system itself) by identifying relevant actors (e.g., “contractor” or “supervisor”).

2. **Identify trust relations**. Defines in what kind of trust relationship actors stand to each other. Trust relations range from low trust classes (e.g., “trust in unknown persons”) to high trust classes (e.g., “trust in the company one is employed at”) and might be uni- or bidirectional.

3. **Identify interactions**. Defines which actors interact with each other and modify shared data. This step might look quite similar to the previous one, but actors that interact with each other might not have to trust each other and vice versa.

In the last step, the system architecture can be derived when all properties are eventually identified. Precisely, the question is answered if blockchain technology is required in the first place and, if so, where and how. Certain actors might already be in a mutual trust relationship that nullifies the advantages of trustless and tamper-proof environments. Adding blockchain technology would thus solely increase development overhead. Other actors, however, might be in somewhat brittle trust relationships that require frequent interactions nonetheless. Communication over the blockchain could be advised.

The third possibility are transitive trust relations. Let T be the homogeneous relation between two participants in set P where trust is established either mutually or by using blockchain-based technologies. Then, participants a and c are in an implicit transitive trust relation if a trusts b and b trusts c. This circumstance is formally expressed in first-order logic:

$$\forall a, b, c \in P : (aTb \wedge bTc) \implies aTc \tag{2.2}$$

Those kinds of relations build the transition points from blockchain-based trustless environments (bTc) to environments with mutual trust properties (such as aTb w.l.o.g.)

[3]Initially developed by Netflix to provide system resiliency

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

using the mediator participant b [WEHG18]. However, which specific kind of blockchain technology (public vs. private or permissioned vs. permissionless, for example) to use and where to employ them highly depends on the domain-specific use cases. Thus, users of the framework should orient themselves towards the goals that stakeholders wish to achieve with the software product (i.e., keep business strategies in mind) and use them to derive architectural tactics using the described process [BSH+20]. It is only a first step towards systematic blockchain-oriented software engineering, but already models an intuitive way of deriving a macro-architectural draft for hybrid apps.

2.2.2 Transactional Patterns

Distributed applications as part of hybrid architectures typically consist of two components: The business logic and the frontend facing the user. The communication between those two components can be modeled in different ways where each introduces certain advantages and disadvantages in either User Experience (UX) or trust. The goal of design patterns, in general, is to provide some (adjustable) solution to non-trivial recurring problems in software design that eventually lead to positive effects, especially regarding non-functional requirements [Gei15a, WZ18]. Transactional patterns are design patterns in the context of DLTs concerned with the communication between the frontend and the blockchain, who verifies transactions, and who pays the transaction fees [WG18].

One common pattern in open source software are the so-called *Self-Generated Transactions*. Users interact directly with the blockchain and the desired EDCC. This is achieved in either of three ways: (1) sending transactions without detours to some blockchain node (e.g., by using the geth client[4]), (2) using some low-level wallet web frontend (e.g., MyEtherWallet[5]) or (3) by using browsers with built-in wallets. This approach requires little to no trust in any abstract system, but requires users to have at least some technical knowledge (especially because this approach is quite error-prone). Furthermore, Application Binary Interfaces (ABIs) must be publicly available. Self-generated transactions are thus quite useful for testing EDCCs during development [WG18]

Self-Confirmed Transactions are an alternative to self-generated transactions. Instead of the users creating the transactions themselves, the frontend application generates them. This requires some trust towards the distributed app, but it is more convenient since users only have to take care of their private key [WG18]. The third pattern are *Delegated Transactions*. Users interact with the frontend without even noticing the usage of blockchains as data storage in the backend. Frontends use an arbitrary technology to communicate with the backend (e.g., REST or gRPC). The backend can then perform off-chain computations and batch update the blockchain state if necessary. This allows for further cost optimizations (because expensive storage and computation can be performed off the blockchain) and does not require users to keep their own wallet [ET17]. An extension to delegated transactions offers the *Meta Transactions* pattern. Users can still

[4] https://geth.ethereum.org/ (accessed on 2022-11-29)
[5] https://myetherwallet.com/ (accessed on 2022-11-29)

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

sign transactions with their private keys if necessary without the need for private wallets. This allows for tracking transactions and offers extended transparency. Both delegated and meta transactions undoubtedly offer the best UX but require a substantial amount of trust in the established infrastructure beforehand. Which transaction technique to use once again highly depends on the use cases, stakeholder requirements, and the target audience of the system [BSH+20].

2.3 Onchain vs Offchain

Blockchains such as Bitcoin and Ethereum aim to transfer value in an environment that does not rely on any centralized trusted third party. This kind of decentralization is achieved by establishing a peer-to-peer network in which each node is a full replication of the entire state of the system, guaranteeing, due to the intrinsic property that each node is incentivized in some form to act trustworthy, that nodes validate one another regularly and punish malicious behavior. In such an environment, however, transactions must be executed on all peer nodes of the network equally. Blockchains that allow custom code to be executed in the form of smart contracts must accommodate the additional execution and storage requirements in some form. Ethereum, for example, associates each operation with a fixed cost, which makes on-chain execution of more complex programs also more expensive [But22].

The limited (and costly) on-chain storage furthermore does not allow developers to store large files like images or videos directly on the blockchain since the cost would be unproportionally high. Blockchains also do not accommodate for any privacy concerns. Any information (even in the form of private variables in smart contracts) can easily be accessed by any participant. Hence, the need for off-chaining storage and computation solutions (sometimes referred to as *blockchain tactics*) was born [WEMG19, ET17].

2.3.1 Smart Contracts

Some blockchain implementations allow developers to write custom applications that are executed enclosed by their trustless and tamper-proof environment. Those applications are typically referred to as smart contracts or EDCCs. Ethereum was one of the first technologies that provided a Turing-complete and tamper-proof programming language by leveraging on their virtual machine with Solidity[6] as their syntax.

Smart contracts[7], however, are limited in their capabilities because they must be deterministic and return the same results on every node of the network that executes the code. This means that reading from HTTP sockets or a local file system, for example, is out of scope. Even the task of producing pseudo-random numbers is a non-trivial challenge in Solidity and has to be implemented with caution, otherwise causing severe security issues. Allowing such operations would immediately render consensus undecidable since

[6] `https://soliditylang.org/` (accessed on 2022-11-29)

[7] Not to confuse with legal contracts.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

every node might produce another output for the same input and sequence of instructions executed (one node times out on the network connection, while another computes the result as expected, for example) [But22].

2.3.2 Offchaining Storage

Blockchains are typically maintained by a larger number of nodes where each node acts as a replication that holds a copy of the chain's current and previous state. This allows all actors to perform operations in a trustless environment but comes at the cost of information being redundantly stored on each node. Hence, a file with N bytes grows linear with the number of nodes. On a chain with M nodes, a total of $N \cdot M$ bytes of disc space is required. Besides being expensive, data on the blockchain is also publicly available to anyone at any time [Nak09, But22].

One might conclude that storing a reference to a large data set which itself is stored on a cloud-native storage solution like Amazons S3[8] or Google Cloud Storage[9] might be sufficient. Doing so, however, would allow users to change the data in the cloud, whereas the reference in the blockchain stays the same. This might be a good approach if trust is not of utmost importance for the participants using the data like, for example, for the images used by the online platform for the CryptoKitties[10] Non-Fungible Token (NFT). In cases where participants are in a conflict of interest and rely on a blockchain's trustless and tamper-proof environment, this naive approach is no longer viable. Thus, one has to find a solution that is manipulation resistant if storage of large amounts of data is required [WEHG18, ET17, XWS19].

One way to achieve manipulation resistance in off-chain storage solutions is to make stored data immutable (i.e., neither allowing manipulation nor deletion). However, this must be enforced by the off-chain storage itself. Another, more convenient way is to use the *Content-Addressable Storage Pattern* as proposed by Eberhardt et al. [ET17]. This pattern aims to provide a solution that allows trustless outsourcing of large amounts of data. This is achieved using so-called content-addressable storage in conjunction with a smart contract. A content-addressable storage is a storage solution that references all its data by a unique address derived from the hash value. Thus, if the content C of some file F has been modified, a new address from $H(C)$ will be derived (given practical collision resistance). A smart contract can now use $H(C)$ to reference the immutable file F and validate if F has changed in a cost-efficient way either off- or on-chain. Off-chain storage solutions must be highly available; otherwise, the system sacrifices its termination property in the long run. Implementations of content-addressable storage solutions that are available and durable are for example the Interplanetary File System [Ben14] and Swarm[11].

[8] `https://aws.amazon.com/s3/` (accessed on 2022-11-29)
[9] `https://cloud.google.com/storage` (accessed on 2022-11-29)
[10] `https://cryptokitties.co/` (accessed on 2022-11-29)
[11] `https://ethersphere.github.io/swarm-home/` (accessed on 2022-11-29)

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

2.3.3 Offchaining Computation

Performing computation on blockchains is similarly expensive as storing data. Blockchains that support EDCCs, as Ethereum does in the form of smart contracts, typically associate a fixed fee to each operation performed in such a contract. Even the most trivial operations such as multiplications or divisions come at some cost (i.e., one pays money to operate a smart contract) [Woo22]. Data that one has to operate on might be private and only available to a single user. This is an inherent problem of blockchains since all the information has to be publicly available to anyone at any time to perform deterministic computations on all nodes respectively. Naive off-chaining approaches, however, will introduce trust problems. The result from the off-chain computation might be faulty, or off-chain systems might send wrong computation results to gain an advantage. Careful consideration of these possibilities is advised, and verification is required (which is the minimum amount of computation that has to be performed on-chain to assure trust) [EH18].

One way to preserve the trustlessness property of blockchains in off-chain computations is the *Challenge Response Pattern*. First proposed by Eberhardt et al. [ET17], this pattern aims to perform expensive computations off-chain while state transitions are validated on-chain. One participant will generate a challenge transaction of some off-chain computation (imagine checking the win conditions for a chess game). Other participants are thereafter incentivized to check if the given challenge is indeed a valid claim. If so, no further transactions must be sent to the blockchain. Some timeout (required to fulfill the termination property) will trigger the on-chain state transition. Thus smart contracts must not rely on on-chain checks themselves. The state transition is canceled if participants send a valid response in time. The layer-2 rollup *Optimism*[12] leverages upon this pattern for example.

The *Off-chain Signatures Pattern* offers an alternative where state transition is performed off-chain and implicitly agreed upon by all participants using their unique signatures. The only purpose of smart contracts in this pattern is to validate the signatures, check if all participants have signed a given transaction, and perform an immediate state change derived directly from the new state information in the transaction. The off-chain signatures pattern thus allows participants to perform as many off-chain peer-to-peer transactions as required without the involvement of the blockchain itself. A state-changing transaction is sent to the smart contract if all participants agree upon the new state. Compared to the challenge response pattern, this pattern requires less blockchain transactions overall, thus leading to a significant cost reduction. However, one has to consider a certain loss of traceability since more transactions are performed off-chain than on-chain [ET17, EH18]. Some implementations of this pattern already exist to cope with the low throughput of modern blockchain technologies [XWS$^+$17]. These include state channel implementations such as the Perun[13] or the Raiden[14] network.

[12] `https://optimism.io/` (accessed on 2022-11-29)
[13] `https://perun.network/` (accessed on 2022-11-29)
[14] `https://raiden.network/` (accessed on 2022-11-29)

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

2.3.4 Offchaining Trust

Other techniques have been proposed to cope with the privacy issues of blockchains in a similar fashion. Hybrid Distributed Applications (DApps) that only partially rely on blockchains depending on the level of trust required for each stakeholder can reduce the amount of transactions required [WEHG18]. Enclave-based off-chain computing is another approach that utilizes Trusted Execution Environments (TEEs) to perform computations in a fully trusted environment that guarantees source code confidentiality and execution integrity. First defined by the OMTP [OMT09] in 2009, primarily for use in mobile devices, they quickly became a promising technology for edge computing and the IoT. Enclave-based computing also allows the usage of confidential information without making it publicly available to the entire blockchain network [FWSZ21].

As a promising alternative to TEEs, privacy concerns can be tackled by leveraging on Zero-Knowledge Proofs (ZKPs). This cryptographic technique allows participants to prove to some third party that they know a specific value D without revealing D. The level of confidence that the participants possess this kind of knowledge is comparable to the confidence hash algorithms provide when showing that a specific piece of information produces a particular hash value. In the context of blockchains, ZKPs can be used to prove that off-chain computations have been performed according to a certain specification without revealing the computation logic itself or to prove that a participant possesses some confidential information without revealing it, for example. In order to generate ZKPs, a zero-knowledge circuit is required, which implementation is very much use case specific. The generation of such proofs is computationally rather expensive, but verifying their correctness, on the other hand, is not [SYZ+21].

A third technology that allows participants to ensure trust to some extent without revealing confidential information is *Homomorphic Encryption*. Homomorphic encryption allows for computation to be performed on encrypted data. Such a system is formally defined as $\xi(x) \odot \xi(y) = \xi(x \odot y)$ where ξ is the encryption function, and $\odot$ is an arbitrary operation in this homomorphic system. One possible scenario for homomorphic encryption is that a piece of information z is encrypted by participant A. Participant B then performs an operation upon the encrypted piece of information $\xi(z)$ resulting in $\xi(\bar{z})$ and participant A then extracts $\bar{z}$ by decryption [JBS+20]. Depending on the specifics of the computational problem, some of the mentioned alternatives might even be combined [EH18].

2.3.5 Keeping a small footprint

Blockchain transactions are expensive operations. Developers should therefore minimize the number of transactions to reduce the overall cost of their application. One option to do so regarding smart contracts is to check the validity of state transitions off-chain and only trigger an on-chain validity check if the off-chain one fails. Of course, such an approach would require other techniques to guarantee trust, such as the *Off-chain Signatures Pattern* or the *Delegated Computation Pattern*. Another option to reduce

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

cost is to reduce the amount of storage required in smart contracts and to optimize for writes instead of reads. This might look counter-intuitive to some developers because it increases the amount of computations performed. However, this approach will move the computation from the expensive blockchain nodes to cheap off-chain systems. In other words: one should not store redundant information but compute derived information locally. This is only possible because reading from the blockchain is feeless compared to writing. These general guidelines are known as *Low Contract Footprint Pattern* [ET17].

2.4 Business Process Management

In recent years, BPM has received more and more attention from both the industrial and computer science communities because it can not only be used to improve business operations and decrease production costs, but it also gives an overview of the broader requirements that a software system has to fulfill and therefore allows better planning to improve scalability and robustness even considering the integration of blockchain as a reliable source of truth. By definition, BPM assumes that every product a company might produce is the result of a series of well-defined activities[15] that are performed in sequence or in parallel; ordering, structuring, and optimizing these activities according to the needs of the company is one of the key concerns of BPM. This is achieved by the concepts and methodologies that BPM provides to support the design and analysis of activities. The resulting series of activities, jointly implementing a business goal, is called a Business Process (BP). BPs typically encapsulate activities that are only performed by a single company or organization. However, interactions with activities that are performed by others are not prohibited and, in certain scenarios, even desirable, as inter-organizational business process management shows [Wes12b]. One might consider the role of a construction company instructed to refurbish the park in the city center as an example of a BP that relies on external activities. The construction company has to gather information about the surrounding environment (historical and protected land sites, water pipes, and electrical infrastructure running underneath or beside the existing park, etc.) from the city council, has to contract city planners to design the new park and has to get in contact with suppliers that deliver required resources on-time. Another example worth mentioning might be a building administrator (responsible for large commercial or residential buildings) who has to contract maintainers for critical facilities like elevators or escalators. Organizing these kinds of activities in a timely and cost-optimized fashion is a typical use case for BPM and shows the need for activities that interact and depend on the activities of other companies, suppliers, and organizations. The upcoming sections introduce notations that are used to create sound models of such BPs and inter-organizational BPs in the form of orchestrations and choreographies that already integrate core concepts for software systems like data dependencies between activities and object life-cycles in artifact-centric BPM[16] [Wes12e].

[15] Later on also referred to as "tasks".

[16] BPM that evolves around the resulting product and the produced artifacts of each activity and sub-process instead of the entire series of activities. This approach might be beneficial when BPs are

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

2.4.1 Business Process Model and Notation

The Business Process Model and Notation (BPMN) is a graphical specification language for business process modeling maintained by the Object Management Group (OMG). It consists of well-defined elements with very specific purposes such as [OMG11, OMG10]:

- **Flow objects**. Activities and events to be performed during the course of a BP. This includes gateways that allow parallel process execution and exclusive process execution.
- **Connecting objects**. Linking flow objects together to indicate transitions from one object (e.g., a gateway) to another object (e.g., an activity). *Connecting objects* also allow associating messages or notes to flow objects.
- **Swim lanes**. Grouping together coherent flow objects and connecting objects. Swim lanes are typically used to depict companies, organizations, or departments in inter-organizational BPs.

The upcoming paragraphs introduce more complex notations of BPMN by example, by building upon the aforementioned BP of a building administrator contracting a maintenance service. This exemplary BP is a simplification of a real-world scenario and was derived from a model created with domain experts. Figure 2.1 shows the simplified workflow of the facility maintainer. Maintenance is only started upon receiving the *message start event* "receive maintenance request". Afterwards, the maintenance is performed, completed and, if accepted by the supervisor, an invoice together with a maintenance report forwarded to all required parties.

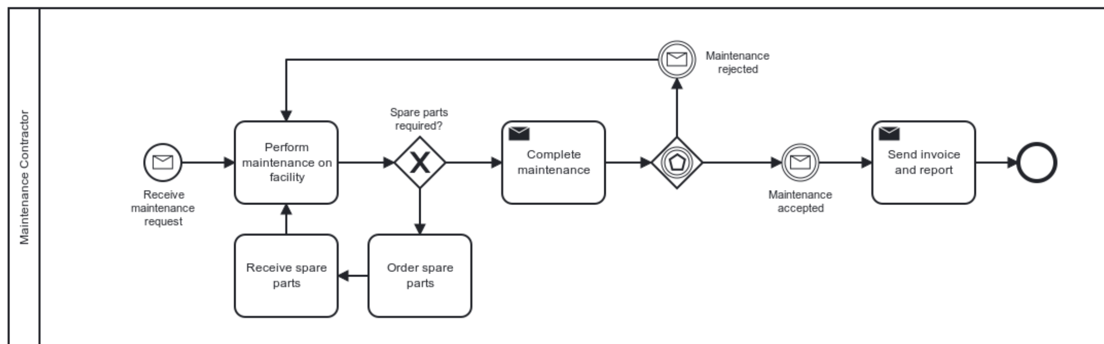

Figure 2.1: A simplified business process of a maintenance contractor.

This BPMN diagram also makes use of some gateways. After the maintainer has performed her first maintenance cycle on the facility, she has to report if any spare parts are required to complete maintenance. If so, spare parts will be ordered; otherwise,

integrated into classical object-oriented software architectures due to the similarities of life-cycles of objects in Object-oriented Programming (OOP) and real-world objects [LSNW20, LWW19].

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

the maintenance is complete. For constructs like these, BPMN introduces the so-called *exclusive* gateway that only allows execution of one of the subsequent activities. In other words, either the spare parts must have been ordered, received, and replaced in the facilities before maintenance can be completed, or no spare parts are required at all. After maintenance has been completed by the maintainer and thus the maintenance contractor, a third party (in this example, the building administrator) can either accept or reject the maintenance. Waiting for activities of third parties to complete is achieved with the *event-based* gateway. This gateway halts further process execution until one of the subsequent events occurs [Wes12b, OMG10]. Single-party BPMN diagrams like these are rarely useful because they do not highlight interactions with third parties and, thus, possible conflicts of interest. Introducing a customer as a building administrator into the diagram in figure 2.1 will improve the example and shows how interactions occur in certain situations. Figure 2.2[17] depicts the full building administrator use case from section 2.4.

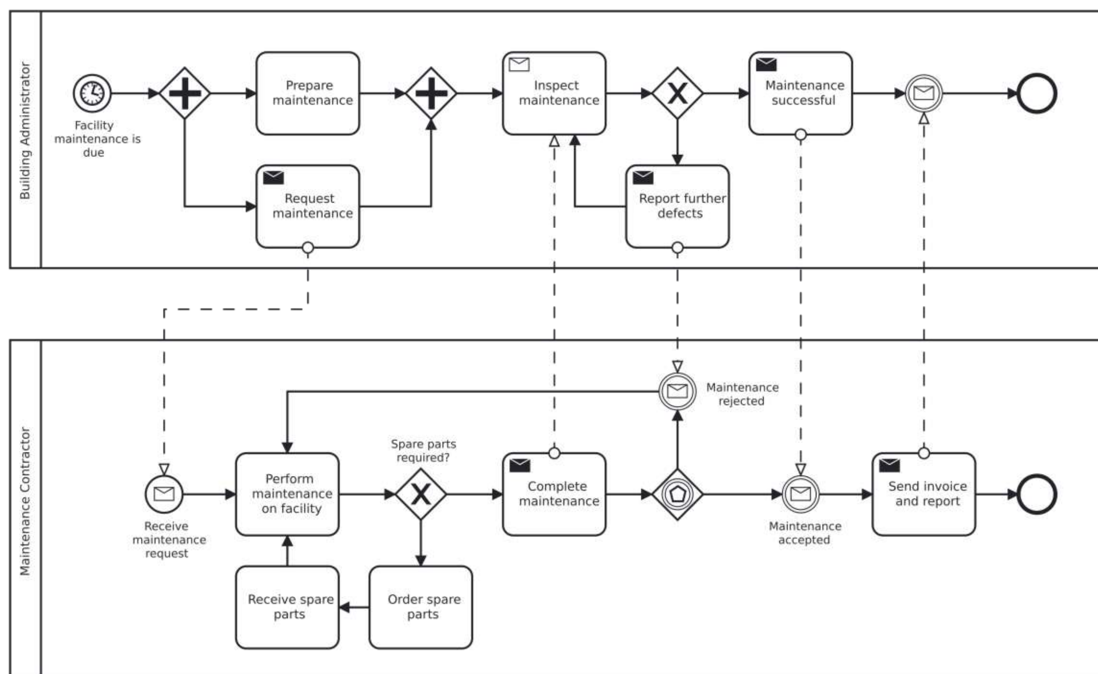

Figure 2.2: More complex two-lane diagram with interactions between participants.

The facility maintenance workflow on the side of the building administrator is triggered by the *timer start event* “facility maintenance is due”. This kind of event indicates that a BP should be started whenever the time constraints associated with the event are met. These might be abstract, as depicted in figure 2.2, or discrete in the form of a timely interval or any other time constraint. When maintenance is triggered, the responsible building administrator has to get in contact with the maintenance contractor and, at

[17]Created with `https://bpmn.io/` (accessed on 2022-11-01)

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

the same time, prepares the facility and the building for the upcoming maintenance and notifies staff and local residents (e.g., by sending a notice that the elevator is not available until maintenance is complete). Parallelization of activities in BPMN is achieved by employing so-called *parallel* or *split* gateways. A *join* gateway is used afterwards to merge all concurrent branches back together if they have been successfully completed. When maintenance is done, the building administrator inspects the maintainer's work and either certifies the maintenance or reports further defects. Once again, this is achieved by employing the *exclusive* gateway of the BPMN notation [OMG11, Wes12b].

2.4.2 Orchestration and Choreography

BPMN introduces a specification language that allows its users to define BPs on a rather granular level using flow and connecting objects. Typically, these objects are grouped into swim lanes where each swim lane defines a single party (e.g., a company or organization). The business process defined for this party is also referred to as *orchestration* or *process orchestration* [OMG11]. Orchestrations are used to provide a more detailed view of activities (and their associated execution constraints) that interact with both external and internal activities [Pel03, Wes12d]. While orchestrations focus on one party's perspective, *choreographies* focus on coordinating interactions between multiple parties. Choreography diagrams that depict a sequence of messages exchanged between parties typically represent a distributed process and activity flow. In other words, the interactions between different orchestrations can be formalized as choreography [AGI+19, Wes12c]. Choreography diagrams are especially useful in business-to-business scenarios where activities of one business have to interact and exchange messages with activities from other businesses [Pel03]. Due to the fact that choreography diagrams are part of the BPMN specification, their notations are also quite similar [OMG11]. Instead of describing activities, choreography diagrams focus on so-called *choreography tasks*. Each choreography task represents the interaction between two or more parties. These tasks are connected with each other by reusing BPMN *connecting objects*. The building administrator and maintenance contractor example is depicted as choreography diagram in figure 2.3.

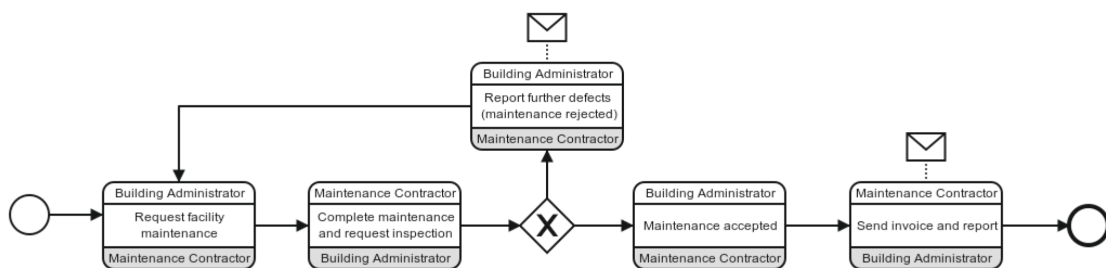

Figure 2.3: Choreography diagram highlighting interactions between two participants.

Graphically, choreography tasks are denoted with boxes with rounded corners. The inside of the box describes the message exchanged, and two bands, one at the top and one at

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

the bottom of the box, represent the involved parties. The band of the initiating party typically uses the same color as the box itself, whereas the darker band is dedicated to the receiving party. Letter-like symbols connected to the bands of the parties attach supplementary information to the message [AGI+19, OMG11]. In this example, an invoice and report are added to the last choreography task before reaching the *end event*.

2.5 Baseline Protocol

The Baseline Protocol, a recently defined Enterprise Ethereum Alliance (EEA) standard and OASIS open source project, tries to enable businesses and organizations to synchronize complex BPs as well as associated data and messages and increase overall system resiliency. The BP introduced in section 2.4.1 is only a simplification of a far more sophisticated scenario. Synchronizing state between counterparties is, therefore, inevitably more difficult than expected. The Baseline Protocol aims to solve this issue by leveraging on the blockchain (or other forms of shared state machines) as an immutable and traceable source of truth that involved participants can trust. Over time, this can increase information security and operational integrity because systems of record no longer have to share potentially confidential data or internal business processes but rely on profound proofs that specific properties of BPs have been fulfilled [EEA22]. The standard, however, is still in a volatile state and can introduce breaking changes at any time. Therefore, the upcoming sections only give a rather broad overview of its concepts.

2.5.1 Architecture and State Synchronization

To ensure "workflow integrity, event ordering, and data consistency", the Baseline Protocol specifies that a compliant implementation must rely on a Consensus Controlled State Machine (CCSM). Blockchains like Ethereum and Bitcoin are implementations of such. The systems of record that the Baseline Protocol Implementation (BPI) communicates with are only loosely coupled with the BPI itself in the form of a standardized API [EEA22]. Figure 2.4 shows a software architecture with a BPI integration.

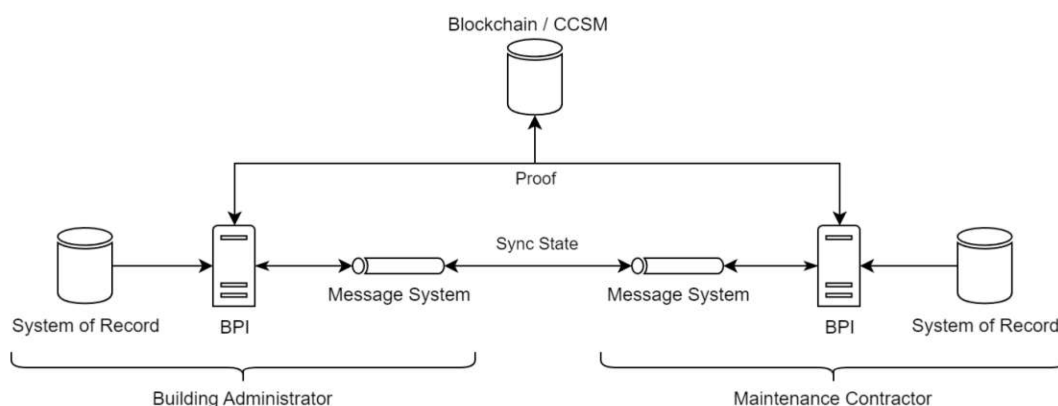

Figure 2.4: Integration of a BPI into the two-party facility maintenance system.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

All involved participants (the building administrator and the maintenance contractor) keep their own system of record that contains the current state of the workflow, required documents, or even confidential data that is not allowed to leave the organization's network. Communication between participants is solely handled by the BPI to exchange necessary information. The BPI generates a proof that the document to be exchanged follows the service-level agreements and stores it on a CCSM to acquire properties such as traceability, immutability, workflow integrity, or data consistency. On the other hand, the data is transmitted entirely off-chain to keep the smallest footprint possible and reduce transaction cost. This ensures that confidential data can be kept private if necessary and allows sending large amounts of data to counterparties without breaking the blockchains' block size limit. Figure 2.5 gives a more in-depth explanation of how communication between counterparties can be realized according to the standard and shows the Baseline Protocol compliant process of the maintenance contractor sending the maintenance report to the building administrator in the form of a flow chart. A system of record that wants to take advantage of the properties of the Baseline Protocol at least has to implement the depicted communication.

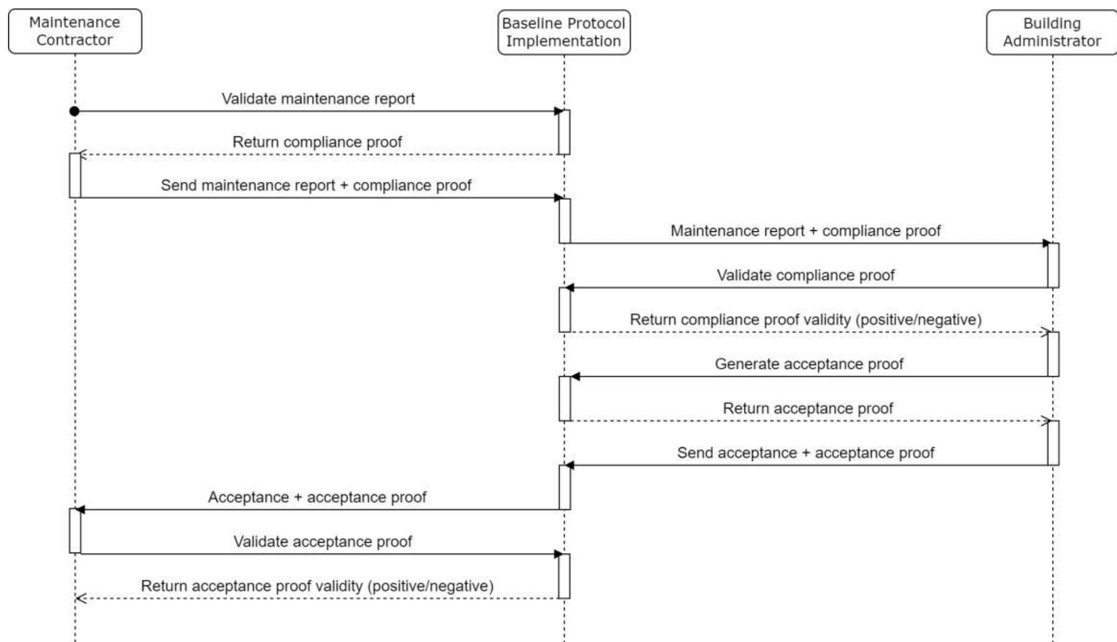

Figure 2.5: Integration of the Baseline Protocol for the exchange of the maintenance report between maintenance contractor and building administrator.

After maintenance has been completed, the maintenance contractor sends the maintenance report to the BPI API to check its validity. This check is implemented using zero-knowledge circuits[18] that precisely portray the service-level agreements that both parties accepted at the start of the BP. If the maintenance report complies, a proof (in the

[18]ZKPs and ZK circuits are out of scope of this work and will not be further discussed.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek WIEN Your knowledge hub

form of zero-knowledge) is returned, and the maintenance report is being transmitted, including the newly generated proof, to the building administrator. Generating proofs, however, is a rather computationally expensive task. Performing it on-chain is therefore unfeasible, considering that hundreds of thousands of proofs must be generated for BPs of large companies and organizations. Thus, layer-2 rollup blockchains like Baseledger[19] must be employed that are optimized to generate such proofs in a more cost-efficient way. Layer-2 rollups like these, again, only store a ZKP of the correctness of all BP proofs on layer-1 CCSMs like Ethereum, to take advantage of the vast amount of participants of such blockchains without exposing any privacy critical data. After receiving the maintenance report with its corresponding compliance proof, the building administrator can now validate the compliance proof to check (1) if the transmitted maintenance report is the one issued by the maintenance contractor and (2) if it is compliant with the previously agreed upon service-level agreement. If the validity check passes, the building administrator can generate a proof that confirms that she accepts the maintenance report in the given form. This proof, once again, can be validated by the other party (in this case, the maintenance contractor) [EEA22].

Even though the Baseline Protocol is still under heavy development and some of its implementation details are out of scope of this work, given its capabilities, it still holds a lot of potential considering inter-organizational BPM and choreographies. Thus, later sections of this work will also investigate possible integrations of the Baseline Protocol into a state machine concept that allows time-traveling verification of BPs.

[19] `https://baseledger.net/` (accessed on 2022-11-01)

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

CHAPTER 3

Related Work

This chapter will briefly discuss the methodology used to conduct the literature review and outlines relevant related work in the field of business process management that utilizes BCT to establish properties such as trust and tamper-proofness. At the end of the chapter, the described concepts will be compared with each other and with the proposed approach of this thesis.

3.1 Literature Review Methodology

The following literature review has been performed using the narrative review methodology with its focus on the state of the art [Str19]. Since BCTs, and especially the utilization of BCTs in the area of BPM, are rather new research topics, only literature of the last 10 years (i.e., 2012 – 2022) is considered relevant. The literature review is used to answer RQ1. Table 3.1 shows a list of primary, secondary and tertiary search terms that were used during the narrative review in the databases of *Elsevier*, *Springer*, *IEEE* and *Researchgate*.

A	B	C
Blockchain	Collaborative Choreography Inter-organizational	Business Process Business Process Management

Table 3.1: Search terms used during the narrative review

The formal definition on how these search terms were combined is shown in equation 3.1.

$$E = [(\bigvee_{1 \leq i \leq 1} A_i) \wedge (\bigvee_{1 \leq i \leq 3} B_i)] \wedge (\bigvee_{1 \leq i \leq 2} C_i) \tag{3.1}$$

Narrative literature reviews do not claim (semi-)completeness on the coverage of a certain topic but give more of a rather unstructured overview of what already exists [RA11]. Systematic literature reviews such as [GGSGL+20] found using equation 3.1 were used as starting point. Furthermore, the snowballing approach was applied to extract primary sources.

3.1.1 Untrusted Business Process Monitoring and Execution Using Blockchain [WXR+16]

The ground-laying work for business process execution utilizing the blockchain published by Weber et al. (regularly cited in subsequent work such as [PSHW20, KJ21, LPGBDW17, LWW19, RCDF20]), aims to find agreements between counterparties on a shared state without the use of a trusted third party. In their exemplary use case, the authors outline a situation where a manufacturer orders supplies via a middleman, and the entire supply chain gets delayed because of production issues on the suppliers' side. Due to the delay, the manufacturer now refuses to accept the supplies from the carrier, which makes the carrier eligible for compensational payments from the supplier or the middleman. To tackle common [PL09], trust-related, supply chain issues like these, Weber et al. suggest the integration of business processes into BCT. Their approach is split into design and run-time decisions. During design-time, a formal model of the business process has to be outlined using a description language similar to BPMN or more blockchain-specific alternatives such as the one described by Ladleif et al. in [LWW19]. This model is then translated into a factory smart contract containing all relevant business process information. The component-based design of this concept allows implementers to use custom translators, such as the one described by Nakamura et al. [NMK18], as well. During run-time, the factory smart contract is used to instantiate and deploy instance contracts that additionally contain information about the roles of the participants (which might not be obvious during design-time). Participants only communicate with each other (and send data) through the blockchain and the deployed instance contracts. These contracts are available in two different forms: (1) a *choreography monitor*, which will check conformance with the business process, and (2) an *active monitor* which is an extension of the *choreography monitor* and can perform additional data transformations or calculations on-chain. Which one to choose depends on the use case and if the data is encrypted. Instance contracts hold and advance the status of the business process and the choreography rules and can trigger external APIs using *triggers*. Triggers are programs that run on full-nodes and listen to smart contract events. Participants will then further process data off-chain that was sent over the blockchain. Due to the payload limitations of transactions on blockchains like Ethereum [Woo22], only smaller payloads might be transferred directly. For larger payloads, just the hash of the data is attached to the transaction, and the data itself is made available using off-chain storage solutions such as Amazon S3[1] or Google Cloud Storage[2].

[1] https://aws.amazon.com/de/s3/ (accessed on 2022-11-29)
[2] https://cloud.google.com/storage (accessed on 2022-11-29)

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Wien Bibliothek Your knowledge hub

However, the on-chain logic and complexity of business processes are limited due to the 24 KB size limit of smart contracts on Ethereum [Woo22]. This issue can be circumnavigated by applying the *diamond pattern*[3], but will inevitably add overall complexity and cost due to the increased amount of smart contracts that have to be deployed. For fairness, the authors suggest that participants may want to split gas money (including smart contract deployments) differently rather than relying on the implicit split of the business process itself. For their prototype, Weber et al. rely on the public Ethereum blockchain. In their benchmarks, the authors ran around 7932 transactions in 32 instances where each business process executed from start to end produced costs of around 0.0347 Ether, which, in 2016, translated to around 0.40 USD but are 109.49 USD at the time of writing this work in 2022. With the fluctuation of prices of cryptocurrencies [PG17, SAW20], conflicts between counterparties might be an aspect to consider when running long-term business processes. Regarding privacy, the authors suggest the usage of asymmetric encryption of data transferred over the blockchain or to rely entirely on permissioned blockchains. Private permissioned blockchains, however, are typically less accepted by industry partners [Bro19], which will lead to centralization and thus nullifies the advantages of blockchains in the first place.

The authors utilized and extended upon their approach to implement a prototypical Business Process Management System (BPMS) as proof-of-concept called Caterpillar[4]. Caterpillar allows the factory and instance smart contract creation from BPMN models and deploys them on the blockchain. When generating smart contracts from BPMN, the system creates an intermediary representation of the business process using Petri nets to allow further optimization of the created smart contracts [GBPDW17]. Furthermore, their system exposes a REST API to make working with business processes more convenient [LPGBDW17, LPGBD+19].

3.1.2 An Architecture for Multi-chain Business Process Choreographies [LFW20]

Leveraging on the aforementioned concept of Weber et al. [WXR+16], Ladleif et al. presented a software architecture that aims to enable the usage of different blockchain technologies (e.g., Ethereum, Hyperledger, Tezos, and others) in a single business process instance. To achieve this goal, the authors only used the blockchain and its smart contracts to store the state and all allowed state transitions of a business process instance. The event and transaction logs provided by most blockchain implementations with smart contract capabilities are used to verify if the Service-Level-Agreements (SLAs) are fulfilled. To enable multi-chain capabilities, the *adapter* software design pattern is utilized where each blockchain, that participants want to integrate, has to implement its own adapter that is able to generate smart contract code, deploy it autonomously and interact with it later on. In most choreographies, not all participants are required to be part of each

[3] `https://eips.ethereum.org/EIPS/eip-2535` (accessed on 2022-11-29)
[4] `https://github.com/orlenyslp/Caterpillar` (accessed on 2022-11-29)

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

individual sub-process. Therefore, the system itself determines which parts are relevant for each participant and only connects to the respective smart contracts and blockchains.

The presented concept gives involved participants more freedom over which blockchain to choose and therefore allows for a more fine-grained adjustment of levels of confidentiality or risk tolerance. Being able to combine different blockchains in a single choreography might be a desirable property depending on the use case and the business needs. Nonetheless, cross-chain communication is still an open problem that might increase the complexity of such a system unnecessarily. Thus, one must consider whether such a system is appropriate for the business process in use.

3.1.3 Runtime verification for business processes utilizing the Bitcoin blockchain [PSHW20, Pry16]

The conceptual model and the prototype presented by Prybila et al. aim to utilize and integrate business processes directly into the Bitcoin blockchain. Therefore, establish trust between counterparties by enabling time-independent verifiability of decentralized choreographies. Their rather flexible approach to this problem allows the process owner to select participants during runtime. This is a useful trait to have considering long-running processes where participants may drop out or get replaced due to changing requirements. At the start of a new business process, the process owner has to select a free Bitcoin transaction output[5] which is used as control token. This control token will be handed over (by using a Bitcoin transaction) to the party that has to perform the next task (i.e., the token holder is fully responsible for the continuation of the business process). In order to enable parallelization of tasks, this concept not only allows process-start, -end, and -handover transactions but also -split and -join transactions. A process-split transaction is performed by the current token holder and contains 1 to N new process tokens in the form of Bitcoin transaction outputs. The new token owners perform their tasks before joining the process back together. It is assumed that a single process-split corresponds to exactly one process-join later on. The authors defined a process-end transaction as a transaction with only a single input which means that a process-join must be performed before a process can complete (if the process was split earlier on). To define the different enriched transaction types, the *data output* and the *Pay-to-Script-Hash* standard Bitcoin transactions are used. The latter of both can carry additional data if necessary. The data will be transferred off-chain, and the transaction itself will only include a hash of the data due to privacy concerns and the fact that Bitcoin is a public permissionless blockchain. Due to the process timestamps and the publicly available data on the blockchain, process owners can monitor the progress of the entire business process and even of single tasks if made available by the current control token owner. The immutability of data on-chain allows the process owner to monitor the progress and verify if predefined SLAs are fulfilled accordingly.

[5]While Bitcoin transaction inputs determine how much value is transferred from one or more addresses, Bitcoin transaction outputs determine to which addresses the value is sent. Typically the input value is larger than the output value. This difference is the block miners reward.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

Due to a median transaction confirmation time of around 7.74 minutes of Bitcoin measured by the authors, only long-running business processes and tasks are viable [PSHW20]. However, their concept also includes a *greedy mode* that allows handover of the control token even if a transaction was not yet confirmed and included in a block. This is because Bitcoin miners accept transactions with references to other, still pending, transactions [Nak09]. *Greedy* transactions, however, pose the threat of being entirely dropped if something goes wrong or malicious participants propose alternative handover transactions. Thus imposing possible consistency issues upon the entire business process and making *greedy mode* practically unviable if trust is of utmost importance. The use of the Bitcoin network itself is also questionable because newer, second-generation blockchains, such as Ethereum, provide Turing-complete scripting languages that can depict requirements directly on-chain and can be used to share digital assets between organizations more easily [KJ21, But22, Woo22]. Also the way data is transferred between participants might pose certain security issues. Once written to the blockchain, the hash cannot be changed and will be available to the public as long as the blockchain itself is available. Even though not a main concern, malicious participants could still abuse this fact at a later point in time and present different data by using collision attacks posing certain security and trust issues. Data once hashed using MD5 or SHA-1 might no longer be reliably verifiable [YJD09, SBK+17].

3.1.4 Blockchain-oriented Inter-organizational Collaboration between Healthcare Providers to Handle the COVID-19 Process [KJ21]

Ilyass El Kassmi and Zahi Jarir presented a conceptual model with a proof-of-concept implementation to handle trust issues when sharing sensitive data in the healthcare sector on a national scale. The authors oriented themselves on a large-scale pandemic use case regarding COVID-19. This concept is especially interesting because it aims to deal with large amounts of data, a changing business process, and privacy concerns. However, it does not focus on the security aspects. To handle ever-changing business processes, the authors use finite state automata modeling and map the model to a Service-oriented Architecture (SOA) to abstract functional and non-functional requirements onto the software architecture. The blockchain, in this case, Hyperledger, is dynamically plugged into the system as a third-party service that introduces a new level of indirection. This Blockchain-as-a-Service (BaaS) approach allows for other blockchains to be used as well and enables a higher level of decentralization since private permissioned blockchains are typically considered to be more centralized since network maintainability is typically ensured by one authority only. The authors introduced the blockchain to depict non-functional requirements, like enabling trust for inter-organizational collaboration only. In this approach, blockchain interactions are seen as a way to fulfill those predefined behavioral non-functional requirements. After processing a workflow step locally, the SOA will forward certain sub-tasks to the blockchain and corresponding smart contracts. Other participants will (automatically) react to those changes in the smart contract and dispatch new events themselves if necessary. In the case of their COVID-19 use

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

case, this might be a PCR test result published by the regional epidemiological center that, depending if the test was positive or negative, is being further processed by the governmental healthcare provider.

Although this novel approach fits this specific use case, it does not generalize well. The authors conclude, that it still heavily depends on the chosen blockchain platform and its limitations in terms of security, scalability, performance, and other key aspects. This might cause issues, especially with short-running business processes. The usage of permissioned blockchains also comes with certain restrictions. Due to the small number of participants compared to public permissionless blockchains, misuse of the blockchain by one participant could render the system untrustworthy because it entirely relies on the integrity of the collaborators [LSP02, SB12]. Hence, the proposed approach assumes a form of common interest in which all members trust each other to at least some extent. Using public permissionless platforms in this context might be viable for use cases where data privacy is not of utmost importance. This, however, cannot be said about healthcare-related data. The authors directly encoded information about the patient and her COVID-19 process into the smart contracts. This means that the information would be publicly available to anybody on a public permissionless blockchain, which contradicts the GDPR[6]. Thus, other storage solutions have to be applied if privacy critical data has to be shared with counterparties.

3.1.5 Blockchain-based controlled information sharing in inter-organizational workflows [RCDF20]

Rondanini et al. inspired their work by the trust problems associated with sharing information in inter-organizational scientific choreographies. Especially when APIs of counterparties and other participants have to be consumed, ensuring that the least privilege principle[7] is being followed, might be of utmost importance. To achieve this goal, the authors solely rely on the distributed consensus algorithm of blockchains to guarantee a correct order of execution of business processes (as shown in [CFR18]). They split their smart contracts into two layers that are tightly coupled with each other: (1) The *coordination layer* that ensures a correct order of execution of the choreography and (2) the *authorization layer* that handles resource access on a per sub-task basis. To be more specific, the authors tackle temporal authorization management (authorization can be revoked) and the least privilege principle with their novel concept. On choreography initialization and smart contract deployment, each participant has to announce, in the form of transactions with a specific payload format, which resources (for example, APIs) they provide. When participants want to start working on a certain sub-task of the business process, they have to send a transaction containing all required resources to the smart contract that handles this sub-task. The smart contract will then check which resources are applicable and grants access. The resource providers listen to the smart contracts event log and will automatically toggle access to their off-chain resources for

[6] `https://gdpr.eu/` (accessed on 2022-11-13)

[7] A participant is only given exactly those privileges needed to complete a task.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

this particular participant. The smart contract will revoke resource access once the sub-task executor announces that their work has been completed. Later on, participants can check what kind of resources counterparties used by examining the log of transactions during execution.

The authors' approach to the least privilege principle and the trust problems associated with choreographies allows for automated authorization management and post-execution verifiability of business processes. However, their concept requires choreographies to be relatively inflexible. Participants have to be assigned to sub-tasks they will work on before the business process even starts. Due to the properties of blockchains and transactions, once revealed, resources cannot be revoked either. Drawbacks like these might have severe consequences, especially for long-running business processes where organizations have to comply with changing legal requirements. Revoking access to resources only off-chain might break the entire business process if other participants still require those resources to complete their sub-tasks.

3.1.6 A Lean Architecture for Blockchain Based Decentralized Process Execution [SSSJ19]

In their work, Sturm et al. try to leverage on the work of Weber et al. [WXR+16] by generalizing the smart contracts in use for a "source-code optimized solution" that is open for extension if necessary. The smart contract is deployed per instance of a business process and only serves as scaffolding. This scaffolding smart contract holds a list of participants and sub-tasks where each sub-task contains a description of the action that has to be performed, the wallet address of the participant that is allowed to complete the task, and a list of predecessor tasks that must be completed before this task can be tackled. After the scaffolding smart contract deployment, the participants and the tasks are added using functions defined in the contract (i.e., blockchain transactions). To partially conform with BPMN, tasks can be defined (1) as simple *TASK*, which only has one predecessor, (2) as *AND*-task, where a list of predecessors has to be completed, or (3) as *OR*-task, where at least one of the predecessor tasks has to be completed before the task itself can be completed. The formal definition of the function $C(t)$ that checks if task t can be completed is depicted in equation 3.2.

$$C(t) = \begin{cases} C(P_t) & \text{for simple tasks} \\ \bigwedge_{t_p \in P_t} C(t_p) & \text{for } AND\text{-tasks} \\ \bigvee_{t_p \in P_t} C(t_p) & \text{for } OR\text{-tasks} \end{cases} \tag{3.2}$$

Where P_t is the set of predecessor tasks and $C(t)$ is defined as recursive function. The authors circumnavigated the recursiveness by adding a *completed* flag to each task to prevent the smart contract from performing redundant computation on-chain. The capabilities of error handling in Solidity, the smart contract programming language of Ethereum, specifically allows participants to communicate with the smart contract more

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

directly. Thus, the authors were able to create a cleaner concept compared to [WXR+16]. Due to the way the scaffolding smart contract works, Sturm et al. proposed, that for their concept, a BPMN translator would not output smart contract code directly, but would generate a list of blockchain transactions that add participants and tasks that depict the business process itself.

This concept tries to solve the lack of trust in choreographies between counterparties by storing as much information on-chain as possible. The authors even mention, that they do not want to rely on off-chain storage solutions. Thus, even large files and datasets would be attached to tasks on-chain. Not only could the transaction and block size limit cause issues, but participants might also be concerned with privacy regarding their data being published. To deal with associated privacy and security issues, the authors recommend the usage of consortium blockchains[8]. However, consortium and private blockchains will inevitably weaken the tamper-proofness to some extent because the voting power is concentrated to preselected nodes. Another disadvantage of this approach is that it relies on the inflexibility of its business processes. Due to the immutability of blockchain transactions [Nak09], tasks and participants cannot be changed in the proposed concept. Even though the scaffolding smart contract could be extended to allow such behavior, the authors did not establish a sound concept of how participants agree to changes in the process.

3.1.7 Interpreted Execution of Business Process Models on Blockchain [LPDGBW19]

Concepts where BPMN models are compiled to smart contracts (as proposed in [WXR+16, KJ21, RCDF20]) do have certain flexibility limitations. To overcome these limitations, López-Pintado et al. proposed a concept that utilizes smart contracts to interpret business processes on the blockchain, similar to the previous work of Sturm et al. [SSSJ19]. To reduce the footprint on the blockchain and allow dynamic modification of the business process during run-time, the authors recommend the usage of a space-optimized tree-like data structure that encapsulates the workflow. Even though the interpreter that encodes the BPMN semantics only has to be deployed once, each sub-process that participants want to add to the choreography will automatically deploy additional smart contracts (one responsible for the sub-process workflow and another one holding associated model data). Therefore, the proposed concept cannot scale well due to the vast amount of associated contracts deployed [LBAG21]. Furthermore, allowing participants to dynamically modify the workflow data structure during run-time might lead to consistency issues (e.g., due to the order in which transactions are accepted by the blockchain [Nak09, Woo22]). Even though the concept enables access restrictions for participants, modification might still lead to inconsistent state. The usage of systems, that should establish trust between counterparties, but might still produce inconsistent state on the blockchain, is questionable. Participants might see different state representations of the same business process, which nullifies the usage of blockchain in the first place.

[8] Blockchains where all miners/validators are predefined and cannot change.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

3.1.8 Decentralized Collaborative Business Process Execution Using Blockchain [LBAG21]

Building upon the work of [SSSJ19] and [LPDGBW19], Loukil et al. aim to further reduce the inflexibility of business process management on the blockchain with their concept. To achieve this goal, three types of smart contracts are deployed in a 3-layer architecture. The central smart contract is called the *interpreter*. It is only deployed once, holds references to the *business process instance* and the *resource instance* smart contracts and implements the facade pattern[9] to access the business process. When a business process is triggered, the interpreter generates a new business process instance smart contract. Participants then dynamically add the BPMN-based configuration to the newly created smart contract through the interpreter using blockchain transactions. Similarly to the business process instance smart contract, the resource instance smart contract that holds and manages the roles of participants, is also generated by the interpreter itself. As mentioned before, these smart contracts are embedded in a 3-layer architecture. The first layer, the *conceptual layer*, is solely responsible for translating BPMN models to an intermediary JSON representation to be more blockchain agnostic. The JSON configuration is then fed into the interpreter smart contract that is part of the *data layer*. The data layer statically encodes a basic business process workflow data structure that is the same for all instances. The routing logic and the process workflow are added to the created business process instance smart contracts in the *flow layer*. Business process instance and resource instance smart contracts can only be accessed and configured by the interpreter. This gives the interpreter more leverage over the entire choreography and enables handling whether or not certain participants are allowed to perform certain tasks or if they are allowed to modify the roles or the business process itself.

The evaluation of the concept shows that business process instance deployment typically leads to higher costs regarding gas compared to compiled approaches such as the one proposed by Weber et al. [WXR+16]. The tight coupling and integration with the blockchain itself also leads to a dependence on the chosen blockchain technology regarding the transaction confirmation time. These two aspects make the concept rather disadvantageous for short-running business processes where many business process instance and resource instance smart contract deployments are required, and state transitions must be performed quickly. Similarly to other proposals, the authors of this concept are also reluctant to tackle the privacy concerns of participants. They advise the usage of consortium blockchains to keep the business process and the associated data private. However, due to the capabilities of the approach to change participants and their roles during run-time, consortium blockchains will inevitably lead to more management overhead since new participants are advised to run their own full node of the blockchain. If participants decline, the usage of blockchain becomes questionable due to security issues like 51% attacks opening up. Nonetheless, the novel architecture presented by the authors gives opportunity for future work to tackle the issues mentioned above, and its loose coupling allows the integration of other approaches as well.

[9] `https://en.wikipedia.org/wiki/Facade_pattern` (accessed on 2022-11-13)

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

3.1.9 Inter-organizational Business Processes Managed by Blockchain [NMK18]

Due to certain limitations of blockchain technologies like throughput, latency, or size restrictions, Nakamura et al. proposed a statechart transformation algorithm for BPMN that allowed the authors to bring down the number of dispatching and receiving events by up to 74% and 65%. Their approach is structured in three subsequent steps. In the first step, a BPMN model is transformed into one statechart for the shared business process on the blockchain and one statechart for each participant. Each statechart represents all states a participant can be in and all corresponding state transitions where state transitions are triggered in the form of events dispatched by any of the participants involved. The authors define a state transition as $e/a_1, \ldots, a_n$ with $e, a \in E$ where E is the set of events that may occur in the business process, e being the receiving event (that starts the state transition) and $a_1, \ldots, a_n$ being the events dispatched after the state transition is completed. Thus, a statechart can be formalized as 5-tuple $\langle S, s_0, F, E, T \rangle$ where S is the set of possible states, $s_0 \in S$ the initial state, $F \subseteq S$ being the set of final states and T is the set of transitions. An event dispatched by the statechart of the shared business process on the blockchain might be used as a state transition starting event for the statechart of one of the participants and thus allows communication between independent statecharts solely relying on a predefined set of events E that all participants share. The proposed algorithm reduces the produced statecharts in the second step. The authors focus on two consecutive state transition where none of them dispatches any events on completion (as depicted in equation 3.3).

$$s_1 \xrightarrow{e_1/\emptyset} s_2 \xrightarrow{e_2/\emptyset} s_3 \tag{3.3}$$

Suppose the first state transition from s_1 to s_2 is removed (including the receiving event e_1 that triggers the transition) and the statechart is rewritten to only allow immediate transitions from s_1 to s_3 using e_2 as starting event (as depicted in equation 3.4). In that case, the observed behavior from outside the statechart will not change because neither less nor more events are being dispatched, and the resulting state is the same.

$$s_1 \xrightarrow{e_2/\emptyset} s_3 \tag{3.4}$$

This transformation will not only allow the removal of the no longer used state s_2 but will also allow the removal of the dispatching event e_1 if and only if no other participant or the shared business process itself requires it. One might even consider a generalization of the transformation algorithm for more than two consecutive state transitions without dispatching events as well. The third step proposed by Nakamura et al. then includes a transformation of the statecharts to smart contracts and automatically generated web user interfaces that allow interaction with the business process.

Even though this novel approach only considers consecutive state transitions and does not include process forks and joins such as the concept proposed by Prybila et al. [PSHW20,

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

Pry16], the algorithm still holds potential for adaptation in future work due to the vast reduction of dispatching events and the extraction of a dedicated statechart representing the interactions with the blockchain.

3.1.10 Data-Driven Process Choreography Execution on the Blockchain: A Focus on Blockchain Data Reusability [LSNW20]

An approach that focuses more on the artifacts produced by a business process and their reusability across instances, was proposed by Lichtenstein et al. The authors' concept focuses on three smart contract types that loosely interact with each other. The first smart contract is the *participants interface*. These interfaces are deployed by each participant involved in the choreography individually and only expose functionality that is relevant for the respective participant to advance the business process instance. The participants' interface further communicates with the *data object store*. This kind of smart contract is deployed once per choreography, holds an instance ID that is increased for each new instance of a business process, and references all required *data object* smart contracts. The data objects are use case and domain-specific smart contracts that depict certain business process artifacts[10]. These data objects are reused throughout multiple business process instances, which will reduce the number of smart contracts deployed per business process instance dramatically compared to other concepts such as the one proposed by Weber et al. [WXR+16] or the one proposed by Ilyass El Kassmi and Zahi Jarir [KJ21]. Data objects are modified through the participants' interface and directly represent the current state of the business process instance. This decoupling of process logic and data objects allows the reuse of data objects in new business process instances and even across choreographies. Therefore, new choreographies with a different process logic and different participant interfaces can import *external* data objects that were produced by other choreographies previously. These external data objects can be used as an additional source of information for newly deployed choreographies.

The dynamic approach proposed is highly beneficial for short-running choreographies where similar business processes are instantiated regularly. Even though the deployment cost is typically higher compared to similar approaches due to the vast amount of data object smart contracts required, the authors could still show a linear decrease in business process instantiation cost. This is due to the circumstance that other approaches deploy all required smart contracts per business process instance, while the approach proposed by Lichtenstein et al., reuses all smart contracts and only has to increase a single variable per instantiation in the data object store. Nonetheless, the approach lacks access rights management and the option to encrypt sensitive data. Thus, further research still has to be conducted considering that data objects are on-chain, and their progression throughout a business process instance is of utmost importance for this approach.

[10]Imagine a car rental business process. In such a scenario, typical data objects would be the car, the driver's license, or the order with its corresponding invoice. Each data object has its own lifecycle with its own custom properties.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

3.1.11 Modeling and Enforcing Blockchain-Based Choreographies [LWW19]

Most of the aforementioned concepts rely on choreography or business process diagrams defined in the BPMN 2.0 standard. However, choreography diagrams are regarded to be purely descriptive and thus lack properties required in model-driven engineering to allow process execution. Moreover, the current standard of BPMN does not reflect most blockchain capabilities directly. Thus, Ladleif et al. proposed a fully backwards-compatible extension to BPMN 2.0 choreography diagrams that enable execution by adapting existing and introducing two new elements that allow the representation of data and logic directly on the blockchain. Data is being stored and shared between participants using two different concepts:

- *Message exchanges*: Messages that are attached to choreography tasks, and are either sent by the task initiator or the task respondent, are represented by blockchain transactions. Due to the circumstance that all transactions are automatically attached to the immutable event log of the blockchain, all participants can verify the state of the choreography at a later point in time. Messages are converted to byte sequences by the client to allow arbitrary data structures being exchanged.
- *Data objects*: The rather artifact-centric approach of this concept introduces so-called *data objects* in the form of variables inside choreography smart contracts. This gives participants more flexibility when sharing information for the progression of the choreography. The append-only data structure that a blockchain is, allows participants to verify if necessary information was shared and if the data objects have been in the correct state to allow a specific state transition, for example.

To ensure correctness of the choreography, smart contracts directly embed the control flow logic derived from the choreography diagram. The extension, however, also allows participants to define custom logic in the form of script tasks. These script tasks can access message logs or data objects to check if a task can be completed or if the next task can be started. To enable authorization management and verify if certain participants are allowed to perform state transitions in the choreography, the authors proposed using an additional smart contract called the *participants registry* where each participant has to be registered prior to choreography instantiation.

Even though the proposed concept performs well regarding correctness of the choreography, traceability, and run-time verification, flexibility is still an issue due to the immutability of smart contracts and, thus, not being able to extend or change data objects once deployed and in use. Even though patterns exist for changing data objects (as proposed by Lichtenstein et al. [LSNW20]), the integration is still an open problem due to the tight coupling of logic, data objects, and smart contracts. Another issue discussed by the authors is the current limitation of smart contracts not being able to trigger tasks automatically, which is a valuable property, given that the choreography relies

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

on recurring tasks such as monthly settlements. These problems and the privacy and confidentiality issues of storing data on-chain still give opportunities for future work and improvements.

3.2 Concept Comparison

The concepts mentioned above are compared with the concept proposed in this work by some of their most distinctive properties in table 3.2. The compared characteristics are as follows: (1) The *execution* type of the concept on the blockchain. Compiled solutions typically generate smart contracts from some descriptive specification language like BPMN or choreography diagrams and automatically deploy them for each instantiation of the BP. On the other hand, interpreted solutions generate scaffolding smart contracts, deploy them once, and get populated through transactions that inject the BP configuration. (2) The *architecture* property characterizes which kind of blockchain is required to allow optimal execution of BPs. (3) The *platform* specifies for which blockchain implementation the concept was proposed and primarily tested on. (4) The *privacy* property categorizes if data privacy is ensured, and if so, lists which mechanism is employed, and (5) often tightly entangled with the previous property, the *on-chain* property indicates what data is stored on the blockchain.

Concept	**Execution**	**Architect.**	**Platform**	**Privacy**	**On-chain**
[WXR+16]	Compiled	Private	Ethereum	Hashed	All
[PSHW20]	Interpreted	Public	Bitcoin	Hashed	Flow
[KJ21]	Compiled	Private	Hyperledger	None	All
[RCDF20]	Compiled	Private	Hyperledger	Authorized	Flow
[SSSJ19]	Interpreted	Consortium	Ethereum	None	Flow
[LPDGBW19]	Interpreted	Private	Ethereum	None	Flow
[LBAG21]	Interpreted	Consortium	Ethereum	None	Flow
[NMK18]	Compiled	Private	Hyperledger	None	Flow
[LSNW20]	Compiled	Private	Ethereum	None	All
[LWW19]	Compiled	Private	Ethereum	None	All
Proposal	Interpreted	Public	Dynamic	Preserved	Dynamic

Table 3.2: Comparison between different conceptual models and their properties

Most of the aforementioned related work, as well as concepts not listed above, heavily rely on BPMN, choreography diagrams, or extensions to one of both to derive their data models and smart contracts. However, because BPMN does not provide any native elements for blockchain integration, some non-functional and functional requirements can not be derived directly. Therefore, some concepts must rely on assumptions that hinder generalization. Nonetheless, recent years have shown a trend from compiled approaches towards interpreted approaches that partially extend existing standards such as BPMN to allow easier integration of trusted third parties in the form of blockchains (an example being [LWW19]). Another interesting observation from table 3.2 is the rather

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

homogeneous distribution of integrated blockchain platforms. Most concepts either rely on Ethereum or Hyperledger Fabric due to their capabilities of allowing distributed code execution in the form of smart contracts. However, these platforms do come with certain limitations. Two of the most important ones are the transaction limit of Ethereum and the privacy issues of data stored on the blockchain in both cases. A lot of the related work mentioned tries to circumnavigate both issues by either employing private or consortium blockchains. Yet, these kinds of architectures have proven less reliable than their public counterparts. The amount of maintenance and setup required, the fact that blockchains still have not fully arrived in the industry and commerce sector, and the lack of experts in this domain led to a decline in the usage of blockchain solutions. Furthermore, private and consortium blockchains are more prone to be attacked due to the overall lower amount of participants[11] [Bro19]. Over time, concepts that become less trustworthy run into privacy and confidentiality issues. Even though some concepts rely on hashing algorithms to ensure some degree of privacy, most of them do not. This might not be an issue if only the state of the overall choreography is shared between participants on the blockchain; it becomes an issue, however, if the orchestration state or data is visible on-chain. Some data or internal BP workflows want to be kept concealed by some participants. Reasons for this can be versatile [PL09]. Thus, rendering concepts that share such data on-chain unacceptable.

3.2.1 Research Question 1

To answer the first research question:

> *What is the state of the art for BCT-based state machines for business process engines?*

Blockchains, and their integration into BPs, are hot topics in the research community right now and are gaining ever more attention due to their huge potential [MWA+18]. However, a lack of awareness for privacy and confidentiality issues is widely present in a lot of concepts and proposals. Even though some ideas arise on how this problem can be solved [CFR18], most related work tackles it by employing private and consortium blockchains. This leads to solutions where the entire state, the BP itself, and sometimes even highly critical data, is kept on the blockchain only to leverage on its traceability and immutability properties. To do so, the state of the art heavily relies on two primary concepts: (1) either by compiling a specification (mostly in the form of BPMN) to smart contracts and deploying them on the blockchain for each instantiation of the BP or by (2) deploying scaffolding smart contracts that are later on saturated with the BP configuration using blockchain transactions. Especially the latter one has gained more attention in recent years due to the cheaper instantiation cost. Both the compiled and

[11] In a three-party consortium or private blockchain, it is enough if the participant with the most computing power, or the most at stake, wants to corrupt the entire network. Scenarios like these, again, require trust between participants and thus nullifies most of the advantages of blockchains [LSP02].

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

the interpreted concepts create a state machine on the blockchain that is as close to the BPMN specification as possible to allow (more or less) direct mappings between both of them. The execution state of the BP, as well as related and shared data (especially in artifact-centric solutions), is stored on the blockchain in most of the concepts and advanced with each process step as described in the related work chapter before.

Nonetheless, a gap was identified in the state of the art that requires a traceable and immutable solution leveraging on the properties and advantages of public blockchains, that ensures data (and partially even internal workflow) privacy and confidentiality. The upcoming sections discuss and propose a new concept for a partially off-chain state machine that builds upon related work and well-established software engineering approaches to fill the abovementioned gap in the state of the art.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

CHAPTER 4

Time-travelling State Machines

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

This chapter briefly introduces the methodology used to derive an applicable concept for a BCT-based Time-travelling State Machine (TTSM) that allows verification of BPs. Furthermore, it discusses why this methodology has been chosen and what kind of tailoring has been applied. Afterwards, the proposed concept, as well as its software architecture, are described in detail and design decisions are discussed. The remainder of this chapter focuses on the implementation of a prototype that is later on used for evaluation of the concept.

4.1 Design Science Methodology

The concept proposed in this work was designed and developed using the design science methodology for information systems research described by Hevner et al. Design science, sometimes referred to as design research, is a constructive research methodology rooted in engineering and the study and evaluation of the artificial. It fundamentally is a problem-solving paradigm that aims to change the existing by creating artifacts such as constructs, models, methods, and instantiations and by providing utility. Produced artifacts are evaluated against metrics and use cases derived from a predefined (organizational) problem space in a so-called *design cycle* [HMPR04]. According to Denning et al. [Den97] and Hevner et al. [HMPR04], design science seeks to

> "create innovations that define the ideas, practices, technical capabilities, and products through which the analysis, design, implementation, management, and use of information systems can be effectively and efficiently accomplished." [HMPR04, p. 76]

The problem space of efficient privacy-preserving BCT-based state machines that allow traceability, as well as the search process paved by related literature and existing business

process engines, makes design science a well-fit methodology to create a novel approach for a TTSM. The following outlines the need for a BCT-based TTSM approach from an industrial and a research-based point of view. Furthermore, for better reproducibility of the results, the upcoming sections give an overview of the tailoring performed to the design science methodology and its guidelines, as described by Hevner et al. [HMPR04], in order to exactly fit the needs of this work.

Guideline 1: Design as an Artifact

This work produces two distinct artifacts during the course of its design science research approach. The first one is the concept proposed for a privacy-preserving BCT-based TTSM that allows verification of (inter-organizational) workflows and business processes in the form of an abstract software architecture and design. The second artifact is the instantiation of the aforementioned model in the form of a prototype system also used for evaluation.

Guideline 2: Problem Relevance

As stated above, recent years have not only shown an ever-growing interest in BCTs alone but also in their usage for managing workflows and BPs due to some favorable characteristics such as fault tolerance and traceability. Nonetheless, limitations of BCTs, like block size limits, transaction limits, cost, privacy concerns, and a lack of experts, hinder businesses from adapting and using such technologies in the long run. However, they still recognize the potential of BCTs to replace trusted third parties in inter-organizational BPs in order to not only reduce capital expenses but also create a mutual trust basis for everyone to equally participate in regardless of size or resources available [MWA+18, VXBP19, PL09, WXR+16, Bro19, EEA22].

Guideline 3: Design Evaluation

In chapter 5, the prototype, and thus the proposed concept, are evaluated against simplified real-world BPs (e.g., the facility maintenance use case) to demonstrate its practical utility. Furthermore, qualitative and quantitative software engineering testing methodologies were employed to allow better reproducibility of results in future work, including analytical, experimental, and descriptive evaluation methods.

Guideline 4: Research Contributions

The research contributions of this work are the design artifacts. This includes the proposed concept for a TTSM introduced in section 4.2 and the prototypical implementation as instantiation in section 4.3. Additionally, future work can rely on the introduced BPs for evaluation.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

Guideline 5: Research Rigor

This work is based upon a formal background in BCTs, distributed systems and state machine replication [SLHK19, Nak09, Woo22, FLP85]. The formal semantics of statecharts allow for sophisticated analysis and evaluation of the proposed concept [NMK18]. BPMN and choreography diagrams are employed as a common basis for BP specification. Furthermore, chapter 3 provides an overview of state-of-the-art knowledge regarding formalism and pragmatism around workflow execution and state machines on the blockchain.

Guideline 6: Design as a Search Process

The BCT-based TTSM for verifiable BPs concept is designed by (1) performing a related work literature review of state machines and workflow execution engines on the blockchain in chapter 3. This step is followed by (2) deriving simplifications of real-world BPs using requirements engineering methodologies such as requirements elicitation [CK92] in the form of problem scoping, understanding, and visualization using BPMN and choreography diagrams. Thereafter, the (3) iterative search for an applicable concept is performed by evaluating different prototypical implementations for their utility against aforementioned BPs, deriving software architecture diagrams to perform architectural analysis as well as using other analytical and experimental methods mentioned later on in chapter 5.

Guideline 7: Communication of Research

The artifacts produced are described in detail in sections 4.2 and 4.3. Additionally, chapter 5 provides context in the form of a simplified real-world scenario that the proposed concept is being evaluated against. This enables technology-oriented audiences to implement and extend upon the proposed concept. Furthermore, the problem statement has been described in detail in chapter 1 to allow management-oriented audiences to determine if organizational resources should be committed.

4.2 Proposed Concept

Based on the aforementioned gap in the state-of-the-art in section 3.2 and the described motivational scenario of a building administrator contracting a facility maintenance service provider in section 2.4.1, a novel approach for a BCT-based TTSM[1] that allows time-travel verification of BPs is proposed by this work. The concept aims to provide a (partially) privacy-preserving state machine that allows the definition and instantiation of workflows and the transition between states of workflows while ensuring consistency and traceability by leveraging BCT. Furthermore, it aims to provide a straightforward interaction mechanism for past workflow states. In other words, participants should be able to easily verify the correctness of a workflow's past states and state transitions.

[1] Later on only referred to as TTSM.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

The main objective of this chapter is to describe a TTSM that enables off-chain workflow execution that smoothly integrates with existing blockchain solutions to make use of some of the properties of BCTs. Additionally, the concept should be integratable into existing systems of record. Before describing the concept in more detail, the goals and non-goals are clarified. This chapter does not aim to describe a concept:

Code	Description	Reason
NG1	for a blockchain, layer-2 rollup or smart contract	BCT
NG2	that itself ensures safety and liveness properties	BCT
NG3	that itself ensures (Byzantine) fault tolerance	BCT
NG4	that itself establishes consensus between participants	BCT
NG5	for validating supplementary rules	OoS
NG6	that determines how data is persisted on the blockchain	OoS
NG7	that translates descriptive diagrams to TTSM configurations	OoS
NG8	that determines and ensures the identities of participants	OoS

Table 4.1: List of non-goals for the proposed concept

Some of the declared non-goals mentioned above originate from properties blockchains ensure, and the TTSM concept only leverages upon (indicated by reason BCT). Other non-goals are simply out of scope (OoS) of this work due to their significant complexity.

Instead of utilizing smart contracts or EDCCs that are directly executed on the blockchain, the proposed concept aims to create an abstraction layer for these kinds of technologies to make the system blockchain agnostic. This allows the usage of the most suitable blockchain for a given workflow. Furthermore, it permits developers to integrate yet-to-be-developed blockchains or layer-2 rollups without being vendor-locked. Not only does this increase the flexibility of possible implementations tremendously, but it also helps mitigate future security issues. In case of a newly discovered security threat in the currently used blockchain, developers can decide at any time whether the blockchain used is still suitable or if they might change to newer versions or entirely other solutions.

This, however, requires the TTSM to keep track of its workflows while still having to provide traceability, immutability, consistency, and persistence properties at the same time. One part of the solution to this challenge is to retain all events that ever occurred in a persistent storage. Thus, an event-driven software architecture is proposed where its event bus is used for communication between modules. Each event dispatched into the event bus is permanently stored. Modules interested in these events subscribe to the event bus and execute their domain-specific logic (e.g., performing validation or creating statistics). The results are either directly fed back into the event bus as events for usage in other modules or kept separated from the rest of the workflow (e.g., in the form of logs inside a logging system[2]).

Figure 4.1 shows a software architecture diagram that visualizes the aforementioned macro software architecture of a TTSM. To allow for a better separation of concerns, the

[2]Using Prometheus or the Elastic Stack, for example (links accessed on 2022-08-14)

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

architecture is split into four modules where each module is fully encapsulated by itself and only loosely coupled with others.

- The *Workflow* module is solely concerned with the semantics of workflow execution, conversion from arbitrary process models to statecharts and optimization of such.
- The *Rules* module enables supplementary (pragmatic) rules that might even span over the entire lifetime of a single workflow instantiation.
- The *Persistence* module permanently stores all workflow commands and events[3], that ever occurred during workflow execution.
- The *Consistency* module communicates with the blockchain and is the only module that also communicates with other participants in the workflow.

Each module materializes its own view of the data dispatched over the event bus and can consist of multiple sub-components if necessary. The architecture shown in figure 4.1 can technically be extended by any number of modules that the specific domain requires.

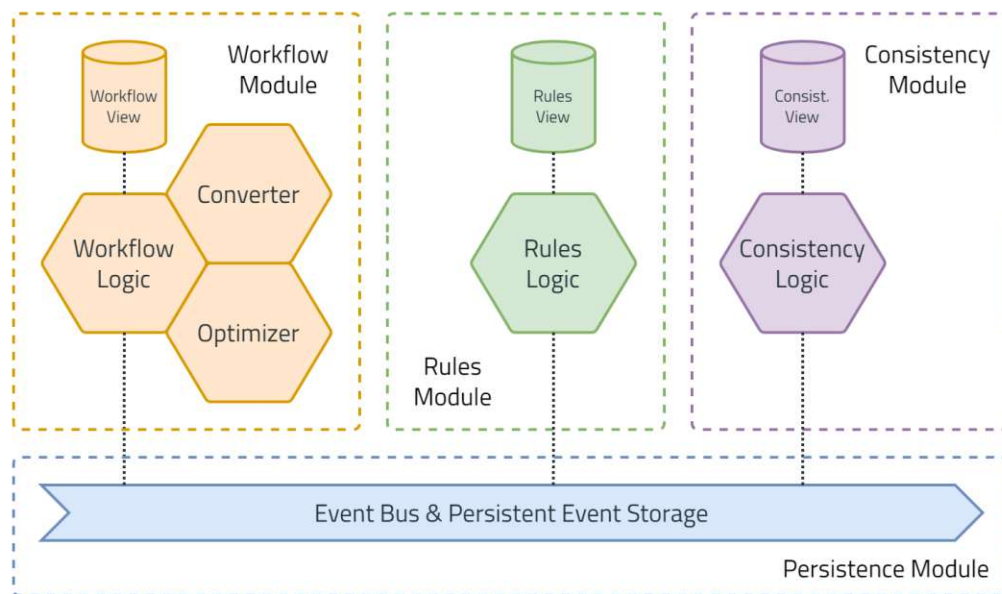

Figure 4.1: Macro architecture and event store design of a TTSM.

In this software architecture, modules only communicate with each other over the event bus using predefined interfaces. It not only tries to provide a clear separation of concerns but also aims to keep stuff that changes together in close proximity to each other to

[3]Commands are actions that the users actively dispatch (e.g., "create workflow"), while events are actions that passively occurred during the execution of a software artifact (e.g., "workflow checked"). Commands are typically in present-tense, while events are in past-tense [Car22].

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

improve maintainability and overall system stability. Larry Constantine, a US software engineer, shaped Constantines law, being that:

> "A structure is stable if cohesion is high, and coupling is low." [New19, p. 16]

To better illustrate the proposed concept, the remainder of this section follows along the life cycle of a single command. Commands include the creation of workflow definitions, the instantiation of workflows, and state transitions. Figure 4.2 outlines the steps that each TTSM command has to successfully pass before consensus between participants can be reached, starting with the syntax and semantic check.

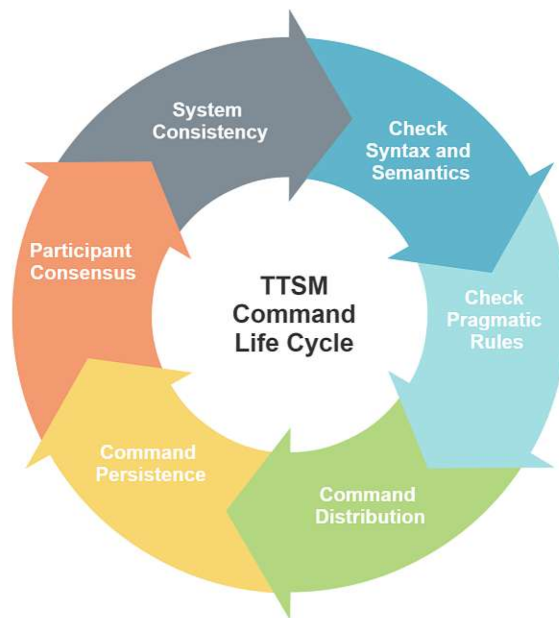

Figure 4.2: Life cycle of a command dispatched in a TTSM.

Similar to figure 4.2, the upcoming subsections begin with the conversion of business processes to statecharts and a check for syntactic and semantic correctness of a command. Afterwards, the rules module verifies if the command passes the employed pragmatic rules. This is followed by the distribution and persistence of the command in each participant's TTSM. Eventually, when all participants checked and accepted the command, consensus is reached, and the system devolves into a consistent state. Note that the command life cycle is similar to the data flow in a TTSM, where a command starts at the workflow module, passes through the rules module, and eventually reaches the consistency module. All while leaving its footprints (in the form of events) on the event bus inside the persistence module with each step. If module internal processes of defining new workflows, instantiating workflows, or performing state transitions diverge from the presented life cycle, this is stated explicitly.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

4.2.1 From workflow models to statecharts

Once the participant's chosen command arrives at the workflow module, it must be converted into a statecharts-compliant format. Statecharts are used as the TTSMs internal representation of workflows due to a vast amount of advantages, compared to Finite-State Machines (FSMs), for example, regarding workflow execution. This not only includes extensions for concurrency[4], but also communication between multiple participants using events [Har87]. Furthermore, statecharts are standardized in [BAA+15], which allows developers to create appropriate tooling[5] and the formal definition makes composition, optimization, and evaluation of statecharts more precise [NMK18]. Additionally, transformation algorithms from UML sequence diagrams to statecharts [ZHJ04] and from process models to statecharts [NMK18] have been proven viable. Given the concurrency properties of statecharts, even BPMN and choreography diagrams can be converted; however, a proof of semantic completeness for these transformations is still outstanding. Given the capabilities of statecharts and the abstraction that this transformation provides, a magnitude of modeling languages can potentially be integrated into a TTSM. For simplicity reasons, and due to the complexity of such transformations, the remainder of this work (including the prototypical implementation in section 4.3) assumes that participants only input statecharts compliant workflows and state[6].

After the transformation, the resulting statecharts are fed into the optimizer. Given a list of optimization algorithms, the TTSM must guarantee that the specified algorithms are executed in order. A simple example of how this could be achieved is given in algorithm 4.1.

Algorithm 4.1: Ensure optimization algorithm order

Input: An un-optimized statechart s, and a list of optimization algorithms O
Output: Optimized statechart s'

1 $s' \leftarrow s$;
2 **forall** $o \in O$ **do**
3 | $s' \leftarrow o(s')$;
4 **end**
5 **return** s';

Optimizers that ignore the given order can cause undesirable side effects because optimization algorithms have no means of being commutative[7]. The optimizer only runs once per workflow definition and always returns valid statecharts. There is no limitation to the amount of optimization performed; however, the semantics of the output s' should always be the same as the semantics of the input s. An example of such an optimization

[4] An essential property for workflow execution because activities can be performed in parallel or even non-deterministic.

[5] e.g., `https://xstate.js.org/` (accessed on 2022-08-15)

[6] Extensions for other formats (like BPMN) are highly encouraged to be part of future work.

[7] i.e., $o_1 \circ o_2 \neq o_2 \circ o_1$

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

algorithm was published by Nakamura et al. [NMK18] (see related work section 3.1.9). Eventually, the optimization step aims to reduce the number of events emitted to the blockchain to reduce overall cost. Optimization algorithms themselves, however, are out of scope of this work. Figure 4.3 shows the described data and process flow inside the workflow module as sequence diagram.

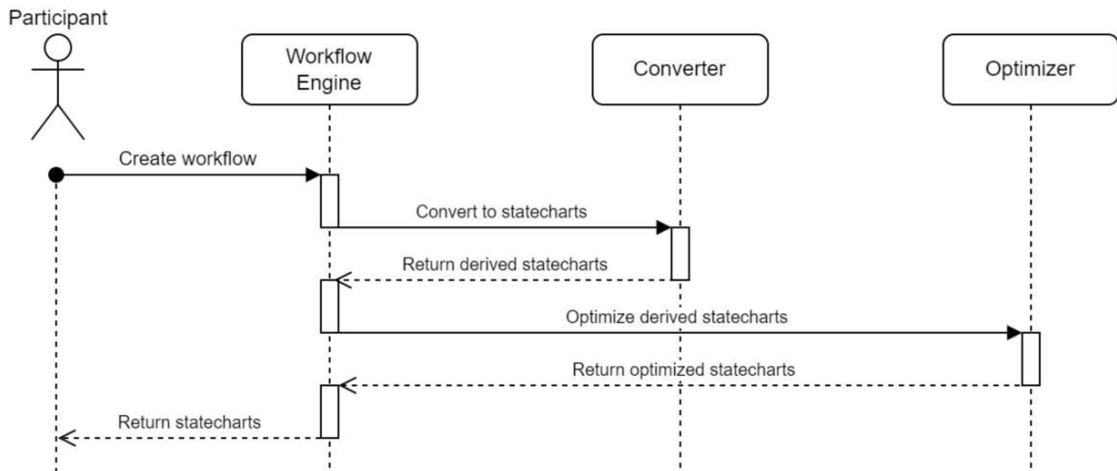

Figure 4.3: Sequence diagram for the creation of a workflow definition inside the workflow module.

Once optimization is complete, the workflow module checks if the given statecharts configuration is valid. Similar semantic checks are employed for workflow instantiations as well as state transitions because the state of the entire system and all workflows are known at any time and exposed by the persistence module. If the command is not allowed in the current state of the system or the workflow, it is rejected. Otherwise, it is forwarded to the event bus.

4.2.2 Distinction of workflows and instantiations

As depicted in figure 4.1, a TTSM is separated into four distinct modules. Clients, to be more precise *workflow participants*, solely interact with the system by talking to the exposed API of the workflow module. This intended abstraction aims to hide implementation details such as the used blockchain in the consistency module or the attached rule services in the rules module. A TTSM workflow module exposes three fundamental commands that participants can trigger at any point in time: (1) participants can propose an entirely new workflow definition with new activities and interactions between participants in the case of a choreography, (2) they can launch a new instance of a previously defined workflow definition and (3) they can initiate a state transition on a previously launched workflow instance. The separation of workflow definitions and workflow instantiations allows the association of data to a specific instantiation and thus improves flexibility [Wes12a]. This is similar behavior to a lot of artifact-centric

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek WIEN Your knowledge hub

approaches such as the ones proposed by Ladleif et al. [LWW19] or Lichtenstein et al. [LSNW20]. Furthermore, the separation of workflow definitions and instantiations has been shown to be rather advantageous compared to other approaches in interpreted BCT-based workflow execution engines due to the smaller footprint left behind on the blockchain itself [SSSJ19, LPDGBW19, LBAG21].

4.2.3 Persisting system state

After the successfully converted and optimized statecharts have been returned to the user for further insight, a single event is dispatched for each action to be performed (defining a new workflow, instantiating a previously defined workflow, and performing state transitions on a workflow instance) to the persistence module. The persistence module stores the entire state transition event, including its payload if present, without further processing. Technically, this allows participants to implement and add custom storage solutions without having to modifying the TTSM itself. For large payloads, for example, state transition events may only want to reference hashes to more domain-specific databases[8] [XWS19, ET17].

Storing state transition events (and commands in general) persistently is of utmost importance for a TTSM. All kinds of events dispatched in the system represent a delta Δ (or difference) that advances a particular workflow state and, therefore, the overall system state[9]. Besides *delta events*, there also exist *fact events* that tell the system in which state a particular sub-state has been at a certain point in time. Unlike *delta events*, *fact events* are not used in TTSMs because they can only be undone by entirely deleting them. On the other hand, when using *delta events*, one can define an inverse operation[10], which allows jumping forwards and backwards in time without modifying the event log itself. Therefore, the sum of commands (including state transitions or workflow instantiations) represents the current state of the system [Bel22, BDM$^+$13].

So-called event-driven software architectures not only allow time travel but also promote loosely coupled sub-systems and enable even better scalability. In other words, dispatched events do not require to be captured and processed by anyone. This allows modules to write events on a more fine-grained level and only consume data that is required. Additionally, due to the already employed event bus system, event sourcing can be used effectively and efficiently to create projections from the stored events and thus derive not only workflow critical information (such as the current state of a workflow instance) but also meta-data around the execution of workflows and the TTSM itself. Furthermore, the event bus allows TTSMs to be split into multiple microservices[11] if necessary. Not

[8] Such as Speckle for Building Information Modelling (BIM) data or the Interplanetary File System (IPFS) [Ben14] as content-addressable storage (links accessed on 2022-08-21)

[9] Imagine a counter that adds one to the current value each time the user presses a button. In this scenario, $+1$ is the Δ to the current state, which in this case, is a counter c that is initialized with 0.

[10] Recall the counter example from before. The inversion of the $+1\Delta$ would be the -1Δ.

[11] One microservice per module.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek WIEN Your knowledge hub

only can this improve scalability and availability, but also maintainability of the entire system dramatically [BDM+13].

Due to the already employed event bus, a TTSM aims to strictly separate between actions that modify the state and actions that solely accumulate and read the state to improve security and flexibility. Therefore, the persistence module embraces the Command-Query Responsibility Segregation (CQRS)[12] pattern as one of its design principles. At its core, CQRS aims to separate database reads from database writes (i.e., a question should not alter the state of the system) [Mey88]. Separating reads from writes further improves scalability because each state transition allows an arbitrary payload with an arbitrary data structure to be attached. This means that the output of the event sourcing stage (i.e., the current or any past state of a workflow instance or the system itself) can permanently be stored using a suitable database technology that is optimized for reads, for example. Event sourcing, combined with CQRS, furthermore enables asynchronous communication with external services. To put this into the perspective of a TTSM, participants can advance the state independently from a blockchain's required block transaction time (which can be up to 20 minutes on some blockchains [Nak09]). Nonetheless, the system must expect rollbacks if workflow instances are advanced without confirmation from the rules or consistency module.

4.2.4 Travelling through time

The persistence module is the part of the system that enables time-travelling due to its event sourcing and CQRS capabilities. If participants want to check if a specific event occurred or a workflow instance has fulfilled a particular property, all they have to do is (a) either search for this specific event by traversing all dispatched events forwards or backwards or (b) replay all previously dispatched events until a certain point in time is reached. The following exemplary list of events is used to better visualize this concept:

E1 (12:00) – Create workflow W

E2 (12:10) – Create workflow instance I of workflow W

E3 (12:20) – Perform state transition from state A to state B on I

E4 (12:30) – Perform state transition from state B to state C on I

If participants want to know, what the state of workflow instance I was at 12:25, they have to replay all events that happened before or precisely at 12:25 in the correct order. Given the example above, this includes E1, E2, and E3. If participants want to know if a state transition from state A to state B happened prior to the state transition from state B to state C, they have to go back in time and check if they can find appropriate events in the expected order (i.e., E3 has to have happened before E4).

[12] `https://en.wikipedia.org/wiki/Command-query_separation` (accessed on 2022-08-21)

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

To allow these kinds of time-travelling capabilities, the TTSM and the persistence module have to ensure *causal ordering* for all events dispatched. This means that some event A has to happen before another event B can even occur - they are *causally ordered* [Bra12]. In the example given above, event E2 can only occur after event E1 has occurred because the creation of a workflow instance requires a workflow to exist in the first place, and thus, they are *causally linked*. A formal representation of this circumstance is given in equation 4.1.

$$A \longrightarrow B \tag{4.1}$$

Since TTSMs ensure *causal ordering*, reliable transitive relations can be derived. The persistence module knows that certain events must occur before others. An example of such a transitive relation of events is given in equation 4.2:

$$(A \longrightarrow B \wedge B \longrightarrow C) \implies A \longrightarrow C \tag{4.2}$$

Hence, one can express the transitive relation of events in a TTSM as a relation T over the set of events E in first-order logic as follows:

$$\forall a, b, c \in E : (aTb \wedge bTc) \implies aTc \tag{4.3}$$

This is an assumption that the TTSM concept and its corresponding persistence module rely upon that allows consistent replaying of events, enhanced rule checking, and more in-depth optimizations before and during workflow execution.

4.2.5 Validating workflow rules

After the persistence module has eventually stored the events, the rules module consumes them. As mentioned above, this module enables participants to create supplementary and often more pragmatic rules for the entire workflow. In other words, when the workflow module checks if state transitions can be performed based on given statecharts, the rules module checks if state transitions can be performed based on previous state transitions and payloads attached. Since the rules module can directly communicate with the persistence module, it can also time-travel and check, for example, if certain state transitions have been performed or if the payload, which was previously attached to one of the state transitions, fulfills certain criteria[13].

As depicted in figure 4.1, the rules module communicates with external rule engines via a predefined interface. This interface allows the registration of rule engines that are triggered as soon as the participant proposes a new workflow definition, workflow instance,

[13]Specifying these criteria is up to the participants themselves, because they are very much domain specific.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

or state transition. The rules defined by the rule engines must all be unconditionally *true*. If one or more rules cannot be checked due to unexpected circumstances (e.g., a network timeout or the rule check returns an incomprehensible result), the rule check is evaluated to *false*. A more formal representation can be found in equation 4.4. If the set of registered rule engines R is empty, the *rules valid* function $RV(t)$ is immediately evaluated to *true*:

$$RV(t) = \top \wedge \bigwedge_{r \in R} check(r, t) \tag{4.4}$$

Each successful, failed, or erroneous response of $check(r, t)$ is re-emitted to the persistence module as a new event to allow for better traceability. This allows the executing participant to better trace configuration or network errors if, for example, one registered rule engine always fails for one specific state transition. After a response has been received from all registered rule engines, another event has to be emitted, which is created by the rules module, to indicate that the action to be performed is indeed allowed according to all previously defined rules. An exemplary algorithm for checking a state transition is provided in 4.2.

Algorithm 4.2: Rules checking algorithm

```
   Input: A state transition t, and an unordered set of registered rule engines RE
 1 a ← ⊤;
 2 forall re ∈ RE do
 3 |   v ← check(re, t);
 4 |   if v is valid then
 5 |   |   emit(valid(re, v, t));
 6 |   else
 7 |   |   a ← ⊥;
 8 |   |   emit(invalid(re, v, t));
 9 |   end
10 end
11 if a = ⊤ then
12 |   emit(all_valid());
13 else
14 |   emit(some_invalid());
15 end
```

In this algorithm, the variable a is used as a tracking variable to flag if at least one rule check failed. If this is the case, the function *emit* writes the event *some_invalid* to the persistence module to inform other modules of the failed rule check and allow better traceability later on. Compared to the optimizer, which is part of the workflow module, the rules module does not have to ensure ordering. This is because the overall validity

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

check of a state transition shown in equation 4.4 is commutative due to the properties of the *logical and* ($\wedge$) operator.

Building upon the definition of the rules module, the number of events emitted per action can be computed using the total amount of rule engines registered $|RE|$. Each rule engine produces exactly one event, and an additional event is dispatched by the rules module after all rule checks have been performed. This circumstance is depicted in equation 4.5:

$$EC_{rl}(RE) = |RE| + 1 \tag{4.5}$$

4.2.6 Sending workflow commands and ensuring consistency

Once the rules have been checked and the appropriate events dispatched, the consistency module consumes these follow-up events. In a TTSM without a rules module (technically possible due to loose coupling), the consistency module might directly consume workflow events. One of the requirements that a consistency module in a TTSM has to fulfill is multi-chain support because of the rapidly changing ecosystem of BCTs. Therefore, it has to provide some abstraction that allows the usage of different consistency strategies. In a TTSM, this might be achieved by relying on the behavioral *strategy design pattern*. This pattern allows the exchange of an algorithm during run-time without changing the interface [Gei15c].

Combined with the approach proposed by Nakamura et al. in [NMK18], where a shared state machine and one for each participant are derived from a given workflow definition, and the approach proposed by Ladleif et al. in [LFW20] for multi-chain support in workflow engines, the consistency module enables the usage of different blockchains not only in different workflow instantiations or definitions but also depending on the payload attached to state transitions, for example. Therefore, if participants agreed upon using different blockchains for different message types prior to workflow execution, the module can choose the applicable strategy implementation during runtime. As an example, the module might rely on an Ethereum-based strategy for larger BIM models, where only the hash of the model and a signature that proves that all relevant participants have seen the model is stored on the blockchain, while it might switch to a Baseledger-based strategy if documents must be proven correct using zero-knowledge. This, however, requires participants to agree on certain BCTs for certain use cases.

If an appropriate strategy has been chosen, the consistency module dispatches the message to all participants involved in the state transition. However, all messages must have at least one participant as recipient. This ensures that the internal workflow activities of one participant are not exposed to other participants. In such a case, the consistency module might even choose a noop[14] strategy to immediately feed back the message to the sender to prevent any network utilization at all and, thus, ensures separation of the shared state machine from the participants one.

[14] no operation

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

Even though the strategy implementations can diverge, the messages exchanged always follow a strict format which is depicted in figure 4.4.

Figure 4.4: Format of a consistency message exchanged by participants.

This rather simplistic configuration is split into two distinct sections: (1) the message header, which contains a unique type identifier in the form of a text string and a commitment reference that references the location where the message is stored (e.g., a ZKP when using the Baseline Protocol, or a transaction reference for Ethereum or Bitcoin). (2) the message data contains an arbitrary payload using an arbitrary data structure. This might be a Base64-encoded file or even just some JSON data, for example. Depending on the message, the payload block might also contain meta information. In the case of a state transition, it stores not only the attached payload but also a unique identifier for the workflow instance on which the state transition is performed upon, the current state, the transition event name, and the expected state. Figure 4.5 shows an exemplary setup of multiple TTSMs to illustrate message flow between participants:

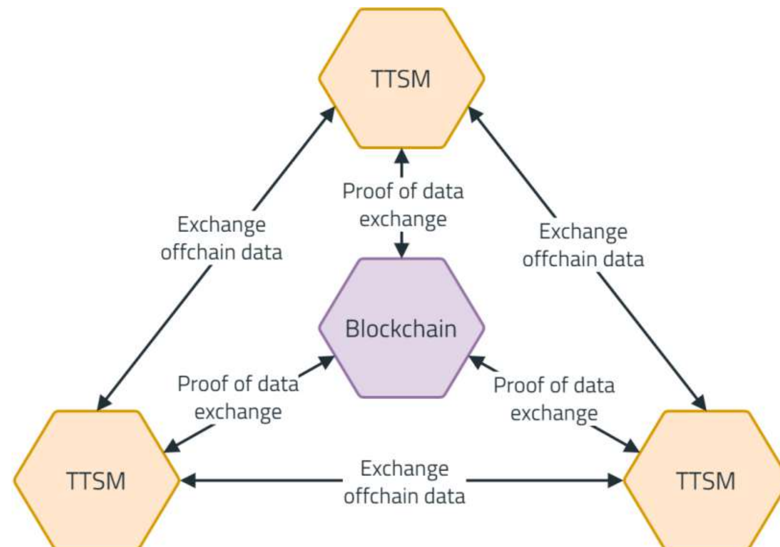

Figure 4.5: Setup of multiple TTSMs with counterparties interacting with each other.

Every participant has her own instance of a TTSM. Each TTSM communicates with other TTSMs by sending messages through the consistency module (by establishing TCP connections, for example). It is also the only module that directly communicates with the blockchain, in order to provide data consistency between participants.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

4.2.7 Processing workflow commands from other participants

After a connection has been established (in one way or the other, depending on the implementation of the consistency module), a message has to be dispatched to all *relevant* participants of a state transition. Participants that receive this message dispatch an internal event containing the payload, the action to be performed on the workflow instance, and the commitment reference (see section 4.2.6). This ensures that all participants have undeniable proof of what happened in which order.

It is important to notice that the internal events of the TTSM used in the persistence module are **NOT** the same events (or messages) dispatched in the consistency module and transmitted (over the blockchain) to other participants. For more complex business processes, this enables that not every single process step (including internal activities) must be written to the blockchain. This is a design constraint of the TTSM proposal. Therefore, there can only be, at most, as many consistency events dispatched as there are persistence events. In other words, each consistency event dispatched must have at least one corresponding internal persistence event being dispatched afterwards. However, not every internal persistence event must be distributed to all other participants. This circumstance is depicted more formally in equation 4.6, where CEC_{total} refers to the total number of consistency events dispatched, and PEC_{total} to the total number of persistence events processed internally by the TTSM.

$$CEC_{total} \leq PEC_{total} \tag{4.6}$$

After receiving the message and converting it into an internal persistence event, each TTSM of each participant has to perform two checks:

- **Syntax and semantics verification**: This check is performed by the workflow module on the transmitted data. This ensures that no incorrect state transitions, workflow definitions, or workflow instantiations can be injected by potentially hostile participants.
- **Rules verification**: The rules module, if present, has to perform a pragmatic verification of not only the transmitted data and its payload but also if the command is allowed in the current context regarding previously performed state transitions and their payloads. As mentioned afore, the rules module has the ability to time-travel to perform these checks.

This two-step process is the same for every command performed, even if it only regards internal workflow activities. Therefore, a participant sending a command to other participants has to perform the syntactic, semantic, and rules verification twice. Once before the consistency module transmits the command to other participants and a second time after receiving her own command. This redundant verification check ensures that commands are correctly written to the blockchain and transmitted over the network.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

Furthermore, it guarantees consistency because all participants work with the same commands and payloads, regardless of being sender or receiver. After verifying the command, participants generate an acceptance or rejection response. This response must be sent to all involved participants.

- **Acceptance**: The participant accepts the command and advances her own TTSM accordingly.
- **Rejection**: The participant rejects the command because either the syntax and semantics, the rules check, or both have failed. Denying a command restores the previous state.

If neither of both responses is sent, other participants cannot further advance their state. Handling these error cases is up to the implementation. Notice that these acceptance or rejection responses do not wait for user input. The response is entirely determined by the TTSM internally. This means that TTSMs do not directly support user-based decisions in workflows (e.g., by asking a user in an appropriate UI if she accepts a state transition given a certain payload). However, this is entirely by design. Such user-based decisions must be modeled as part of the workflow definition itself. Statecharts, and other modeling languages that can be transformed to statecharts (such as BPMN, for example), already include decision elements as an integral part of their specification. Therefore, user input must be handled purely declaratively on the workflow definition level by the participants themselves and is not the responsibility of the TTSM. In other words, if a document must be shared or a decision has to be made, an applicable gateway or task must be added to the workflow.

4.2.8 Eventually reaching consensus

Once a response has been sent by an involved participant[15], she has to wait until all other participants have sent their responses as well. A command that concerns N participants produces at most N responses. At this point, the TTSM has to handle one of three scenarios:

- **Acceptance from all participants involved**: All participants accepted the command and advanced their state accordingly. Thus, consensus has been reached, and the execution of the command is complete.
- **Rejection from at least one participant involved**: One or more participants rejected the command and restored their previous state. Thus, all other participants must also restore their previous state to assure consistency across all parties. In other words, a rejection from at least one participant rolls back the entire command for all participants involved.

[15]Notice that the distinction between command sender and receiver is no longer relevant at this point.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

- **No response from at least one participant involved**[16]: This work is not concerned with this case in particular because it highly depends on implementation details and non-functional requirements such as availability, reliability, resiliency, or fault tolerance, for example.

Given these three scenarios, especially the second one is prone to attacks from hostile participants because they can deny a command infinitely many times. This threat, however, is more of a theoretical attack. Similar to the Nothing-at-Stake attack described in section 2.1.3, hostile participants lose more than their potential victims. Other participants are fully aware of who rejects and who accepts commands due to the strong traceability properties of TTSMs. Thus, the credibility of these participants suffers in the long term.

The algorithm of the consistency module, which determines the outcome of a command, is rather similar to algorithm 4.2 described in section 4.2.5 used to determine if a command passes the rules module. The TTSM-consensus algorithm 4.3 is concerned with the first two scenarios described above:

Algorithm 4.3: TTSM-consensus algorithm

Input: A command c, and the number of involved participants N

```
1  a ← ⊤;
2  forall i ← 1 to N do
3  |   r ← next_response();
4  |   if r is acceptance then
5  |   |   emit(accepted(c, r));
6  |   else
7  |   |   emit(rejected(c, r));
8  |   |   if a = ⊤ then
9  |   |   |   emit(some_rejected());
10 |   |   end
11 |   |   a ← ⊥;
12 |   end
13 end
14 if a = ⊤ then
15 |   emit(all_accepted());
16 end
```

Once again, a is used as a tracking variable to flag if at least one participant rejected the command. In this case, two internal events are emitted: (1) a *rejected* event that contains not only the commitment reference but also the participant and the command itself and (2) a *some_rejected* event that is immediately dispatched afterwards to indicate that the

[16]See FLP impossibility result in section 2.1.9.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

command must be rolled back and the previous state restored. The algorithm, however, does not stop after one participant rejected the command. This is because all rejection responses should also be stored locally as internal events in the persistence module for better traceability.

TTSMs leverage on the concept of “soft state”, which means that data might be inconsistent between participants for a certain amount of time but eventually reaches consistency when things have settled [Vog09]. This is due to the handling of participants that do not respond immediately and the block transaction time of BCTs. However, after all involved participants accepted the command, consensus has been reached and consistency archived. Therefore, the order of commands is typically determined by the consensus algorithm of the chosen BCT. Even though TTSMs have to deal with soft state and rollbacks of commands, this single source of truth takes care of command ordering and establishing consistency between all participants.

4.3 Prototype Design

In contrast to the previous section 4.2, this section describes a practical implementation of the proposed concept that is used for evaluation later on. Notice that not all functionalities that are theoretically possible in a TTSM are implemented to their full extent in this prototype because it only serves as proof of concept. The prototype described uses state-of-the-art technologies and well-established industry standards to show that it is possible to implement a fully functional TTSM leveraging on existing know-how. The upcoming sections describe a potential implementation of each module of a TTSM. This diverges from how section 4.2 described the overall concept because this section focuses more on implementation details compared to the interoperability of the individual modules. The prototype described here is available on Zenodo[17] and GitHub[18]. It makes use of the following technologies:

- **Node.js 16.14**[19]: A runtime for JavaScript, built on top of the V8 JavaScript engine. It is used as an execution environment for the TTSM itself because its event-based design makes it a good fit for systems that heavily communicate asynchronously with other systems.
- **npm 7.8.0**[20]: A package manager for JavaScript and Node.js that handles dependencies using a centralized registry. Furthermore, npm also supports scripting of smaller tasks and a clear distinction between dependencies required during runtime and dependencies required during development. This allows bundled artifacts to be smaller and execute faster.

[17] 10.5281/zenodo.7375788 (accessed on 2022-11-29)
[18] https://github.com/danielkleebinder/ttsm-prototype (accessed on 2022-11-29)
[19] https://nodejs.org/ (accessed on 2022-11-29)
[20] https://npmjs.com/ (accessed on 2022-11-29)

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Wien Bibliothek Your knowledge hub

- **TypeScript 4.7.4**[21]: A strongly typed programming language and superset of JavaScript developed by Microsoft. It is a well-established industry standard, and its sophisticated type system enables developers to create stable software for complex problems.
- **NestJS 8.4.7**[22]: A Node.js framework that leverages on TypeScript for server-side applications that aims to be scalable and reliable. The frameworks module system allows dependencies between modules to be explicitly specified and, thus, creates a clear separation of concerns. This mechanism is heavily made use of to separate modules from each other.
- **XState 4.31.1**[23]: A popular, lightweight FSM and statecharts library for TypeScript and JavaScript. It is used internally to represent statecharts derived from workflow definitions and to perform stateless state transitions in workflow instances. Statelessness is a much-wanted property given the time-travelling capabilities of a TTSM.
- **EventStoreDB 21.6.0**[24]: An event-based database that enables event sourcing. Event databases like these are a perfect fit for TTSMs because they are not only scalable due to their built-in command-query separation and immutable state, but they also support time-travel based on previously dispatched events (see section 4.2.3 for more details on why event sourcing and CQRS are used as core concepts in TTSMs).
- **Docker 20.10.13**[25]: A software containerization technology, that allows the TTSM-prototype to be launched on any system.

The upcoming sections describe, in detail, how these technologies are used to create a state-of-the-art TTSM, based on the concept proposed in section 4.2.

4.3.1 Workflow Module

The structure of the workflow module, and all modules in general, is determined by an internal three-layer software architecture, where (1) the *presentation layer* exposes the endpoints that users interact with to dispatch commands such as creating new workflow definitions or performing state transitions on existing workflow instances. (2) The *application layer* performs the systems workflow logic, such as converting BPMN workflow definitions to statecharts or optimizing statecharts. And, (3) the *data layer* which is responsible for storing all processed information in an immutable and append-only event log. Figure 4.6 depicts the software architecture of the workflow module as

[21] https://typescriptlang.org/ (accessed on 2022-11-29)
[22] https://nestjs.com/ (accessed on 2022-11-29)
[23] https://xstate.js.org/ (accessed on 2022-11-29)
[24] https://eventstore.com/ (accessed on 2022-11-29)
[25] https://docker.com/ (accessed on 2022-11-29)

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

Unified Modeling Language (UML) class diagram. The presentation layer involves the *WorkflowEndpoints* class that uses the NestJS runtime environment to expose HTTP REST endpoints using distinct Data Transfer Objects (DTOs)[26], that are only available in the presentation layer itself. The application layer exposes the *WorkflowService* that relies on the *ConverterService* and the *OptimizerService* to generate TTSM-compliant statecharts. Finally, the data layer persists the produced internal events by directly accessing the *EventStore*. Notice that the arrows are unidirectional between these three layers. This is a design decision that prevents circular dependencies and enables loose coupling.

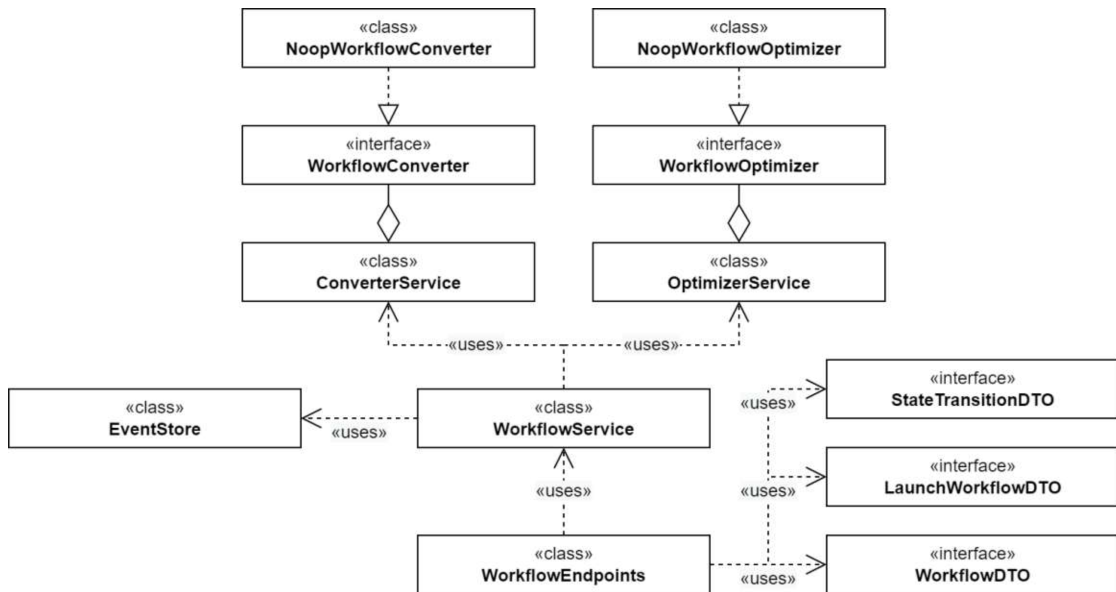

Figure 4.6: UML class diagram of the workflow module of a prototypical TTSM.

The *WorkflowEndpoints* class also generates a fully fetched OpenAPI definition that is rendered with *Redoc*[27], a web-based tool, that creates beautiful single page applications that group and describe exposed endpoints in detail. Generally speaking, the *WorkflowEndpoints* class receives HTTP REST commands with endpoint-specific payloads in the form of DTOs[28]. These DTOs are then converted and fed forward to the *WorkflowService*. Employing such a conversion allows the definition of multiple, independent endpoint classes, where each endpoint class might satisfy different specifications. This further simplifies the integration of existing solutions since the *WorkflowService* class

[26]Used to decouple the presentation from the application layer and only expose data that the API consumer requires.

[27]`https://github.com/Redocly/redoc` (accessed on 2022-10-29)

[28]Notice that all DTOs are marked as interfaces in figure 4.6. This is a peculiarity of the chosen programming language, TypeScript. After compiling TypeScript to JavaScript, the interfaces no longer exist because JavaScript is a dynamically typed language, and TypeScript, in contrast, is a statically typed one. Using interfaces instead of classes is, therefore, an optimization that reduces the final application size.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

independently works with entities instead of DTOs. The endpoints exposed by the *WorkflowEndpoints* class, which allow the definition of new workflows, are listed and briefly described in table 4.2.

Method	**Path**	**Description**
POST	/workflows	Creates a new workflow and its statecharts.
GET	/workflows	Returns all workflows.
GET	/workflows/{id}	Returns the workflow with the given ID.
POST	/workflows/{id}/launch	Launches a new instance of a workflow.

Table 4.2: List of workflow definition endpoints

Based on the concept of separation between workflow definitions and workflow instantiations in section 4.2.2, workflows must be defined before they can be launched. All *GET* endpoints also allow time travel by supporting an additional query parameter called *until* that must be supplied with an ISO-based timestamp [ISO04]. With this utility, participants can trace a workflow definition's progress, for example. Similar endpoints are exposed for workflow instantiations listed in table 4.3.

Method	**Path**	**Description**
GET	/instances	Returns all instances.
GET	/instances/{id}	Returns the instance with the given ID.
GET	/instances/{id}/payloads	Returns all payloads of a particular instance.
POST	/instances/{id}/advance	Performs a state transition.

Table 4.3: List of workflow instance endpoints

After a workflow definition is launched, a workflow instantiation is created. This workflow instantiation, and its progress, can also be queried using time travel. To further advance a workflow instantiation (i.e., to perform a state transition), the participant calls the *advance* endpoint on the appropriate instance.

Newly created workflow definitions, instantiations, or state transitions, are then forwarded to the *WorkflowService*. This service contains the actual logic of the workflow module. Since, technically, it should be possible to not only allow statecharts for workflow definitions but also BPMN or choreography diagrams, the *WorkflowService* has to choose an applicable converter. Once the workflow definition was forwarded through the *WorkflowService* to the *ConverterService*, the *ConverterService* determines[29] in which format the workflow is defined and selects the correct converter strategy. The converter then outputs statecharts, where semantics are the same as the ones previously defined by the participant. Therefore, the prototype internally only ever has to work with statecharts rather than supporting a heterogeneous set of workflow definition languages. Even though it has been proven that transformations from different process definition models to statecharts are possible [ZHJ04, NMK18], this prototype only implements

[29]By reading a flag that participants have to add to the workflow definition configuration.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

the *NoopWorkflowConverter* strategy that inputs statecharts and immediately, without any further transformations, outputs it. This is due to the complexity associated with converter algorithms; however, it shows the feasibility that different strategies could be supported. The algorithm for determining which converter to use is given in algorithm 4.4.

Algorithm 4.4: Choosing a converter strategy

Input: A workflow definition configuration w, and a list of converters C
1 **forall** $c \in C$ **do**
2   **if** $w_{type} = c_{type}$ **then**
3   **return** $c(w)$;
4   **end**
5 **end**
6 **return** *null;*

In the algorithm implemented, c_{type} represents the supported input type of the converter, and w_{type} is the type specified by the participant for this particular workflow definition. After the conversion is complete, the *WorkflowService* forwards the generated statecharts to the *OptimizerService*. This service then picks the appropriate optimizer strategies, which are also part of the workflow definition configuration. However, compared to the *ConverterService*, the *OptimizerService* might not only perform a single optimization algorithm but multiple in a specified order (total ordering is guaranteed by the proposed TTSM concept). The algorithm for this process is defined in the concept in section 4.2.1 in algorithm 4.1. An exemplary definition of a simple pedestrian traffic light workflow with appropriate configuration is given in listing 4.1.

```
{
  "config": {
    "optimizer": ["noop"],
    "type": "STATECHARTS"
  },
  "workflow": {
    "initial": "green",
    "states": {
      "green": { "on": { "TIMER": "red" } },
      "red": { "on": { "TIMER": "green" } }
    }
  }
}
```

Listing 4.1: Exemplary workflow definition of a pedestrian traffic light

The prototype only supports the *NoopWorkflowOptimizer*. It immediately returns the input statecharts without modifications to show the feasibility of this multi-optimizer approach. Similar to the converter, optimizers are also rather complex and, therefore, out of scope of this work.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

The generated statecharts are then fed into the *XState* library to verify their syntactic and semantic validity. The library is used to depict the current state of a workflow instance and enables “stateless” state transitions. In the context of a TTSM, this means that the workflow module only has to store the workflow model in the form of statecharts, as generated by the converter and the optimizer, and the current state of the workflow instance. Note that this “statelessness” also allows easy reconstruction of previous state machines by time-travelling and extracting the state at another point in time. If a participant wants to perform a state transition, the state machine is reconstructed by the TTSM by feeding the workflow model and the current state into *XState*. If the state transition on the reconstructed state machine fails, the workflow module responds with an error.

After conversion, optimization, and validation of workflow definitions, instantiations, and state transitions, the workflow module, and in this case the *WorkflowService* in particular, dispatches a single follow-up event to the persistence module by directly connecting to the event bus of *EventStoreDB*. The list of events is given below:

- **Client.Workflow.Propose**: The participant wants to propose a new workflow definition for the entire network. This event includes the converted and optimized statecharts.
- **Client.Instance.Launch**: The participant wants to launch a new instance of a previously proposed and accepted workflow definition.
- **Client.Instance.Advance**: The participant wants to perform a state transition on a previously launched and accepted workflow instance.
- **Client.Instance.TransitionAccepted**: Some participant wants to perform a state transition, and the state machine has already incorporated it locally.
- **Client.Instance.TransitionFailed**: Some participant wants to perform a state transition, but the local state machine rejects it due to syntactic or semantic errors.

The connection to the event bus is also used inside the *WorkflowService* to create projections of the dispatched events. These projections accumulate all events until a certain point in time (thus enabling time travel) and return the state of a particular workflow definition or instantiation.

4.3.2 Persistence Module

The persistence module, as mentioned afore, is an implicit module that is created by the external *EventStoreDB* system. Modules that need to communicate with the event bus or the event store directly connect using the provided *EventStoreDB Client*[30]. The client

[30] `https://npmjs.com/package/@eventstore/db-client` used in version 3.4.0 (accessed on 2022-11-29)

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

comes with full TypeScript type support and exposes a list of database command and query functions that are executed using gRPC[31], an open source framework for remote procedure calls that is optimized for performance and throughput. To write new events to the event bus, the client provides the *appendToStream* function. This function takes the name of the stream and a JSON-based payload (the event) that should be dispatched. The prototype generates an entirely new stream for each newly defined workflow definition with the name *workflows.{id}*, to which associated events are appended. This increases performance because the overall stream size is reduced and makes reading and generating projections easier. The system does not have to filter for events with specific workflow IDs but, instead, can consume the entire stream at once. A similar mechanism has been employed for workflow instantiations. The prototype generates a new stream with the name *instances.{id}* for each newly launched workflow instance. The IDs used for workflow definitions and instantiations are generated on the client side using UUIDv4.

Another important concept that the TTSM-prototype heavily relies on are projections. To create projections, the *EventStoreDB* client exposes the function *createProjection* that inputs a unique projection name and the projection source code in the form of simplified JavaScript. Projections are typically executed for each new event dispatched to the stream it relies upon. These events are then accumulated[32] over time and stored in a so-called *materialized view*. Figure 4.7 shows the basic architecture and interaction between an external connector and the *EventStoreDB* itself. The unidirectional arrows denote message flow direction.

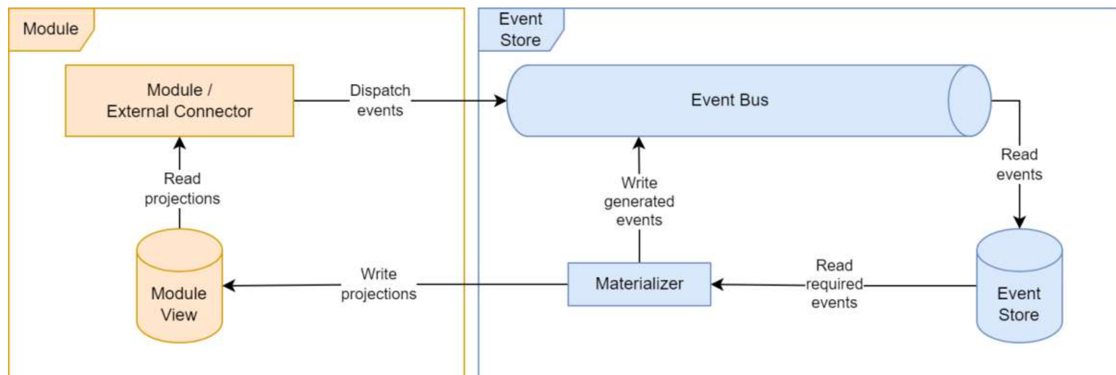

Figure 4.7: Persistence module and event store architecture.

The event store reads and persists all events dispatched to the event bus and its streams. Since projections are executed on the database side as part of the materializer, events can be directly read from the event store without any network delay involved. Therefore, this prototype could technically be scaled horizontally for large-scale systems by increasing the number of *EventStoreDB* nodes. On the client side, projections generated by the

[31] `https://grpc.io/` (accessed on 2022-09-06)

[32] Imagine an event called *increaseByOne* that is dispatched by some client on an irregular basis. A projection that should count the total amount would then be initialized with value 0 and adds 1 to the current value each time this event has been recognized.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

materializer can be queried directly using functions like *getProjectionResult* or subscribed to get notified each time the projection is updated. The second approach is typically the preferred way for event-sourcing systems because projections can be stored directly in appropriate database systems. For example, this might be a relational database system for workflow definitions or an object storage for large data sets like BIM models. For simplicity reasons, the prototype queries projection results only on demand.

In case a participant, the rules module, or any other potential client needs to verify a previous state, time travel is performed locally in the TTSM by accumulating all events from the beginning until a given point in time. This is only possible because the TTSM concept guarantees certain properties such as total and causal ordering of events (see section 4.2.3), or separation between the persistence of the event history and the current TTSM-state [Pry19].

4.3.3 Rules Module

This module is similar in structure to the workflow module described in section 4.3.1. It uses a three-layer architecture, namely, presentation, application, and data layer. The presentation layer exposes the endpoints used to register external rule validation engines, the application layer performs the asynchronous validation whenever new workflow definitions, workflow instances, or state transitions are received, and the data layer, which persists validation results. These three layers are, once again, loosely coupled by the dependency injection mechanism provided by NestJS. Figure 4.8 shows the UML class diagram of the rules module.

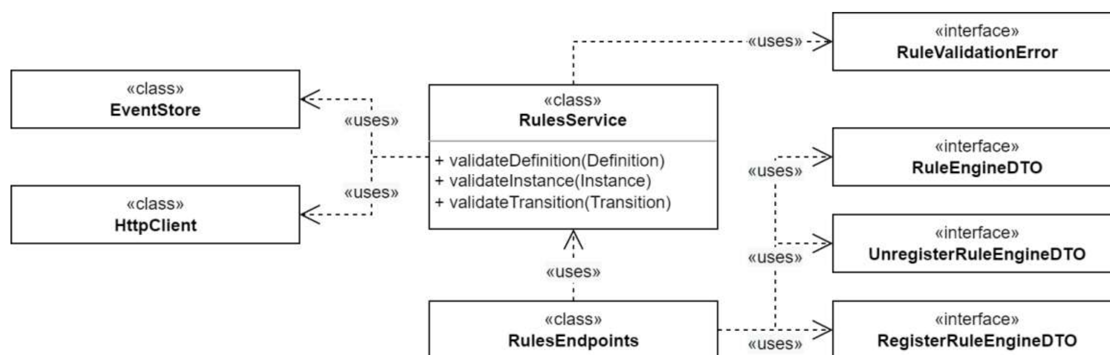

Figure 4.8: UML class diagram of the rules module of a prototypical TTSM.

The *RulesEndpoints* class allows a participant that runs a particular instance of the TTSM prototype to register rule validation engines that are later on invoked by the *RulesService* if necessary. These engines are used to validate and verify if workflow definitions can be created, workflow instantiations can be launched, and state transitions can be performed under a predefined set of constraints called “rules”. Furthermore, systems like these are technically able to connect to the persistence module directly and the endpoints exposed by the *EventStoreDB* described in section 4.3.2. This enables time-travelling capabilities

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

inside rule validation engines. Thus, state transition and their payloads, for example, can be verified against previous state transitions and payloads[33]. The implementation of such rule validation engines and how these kinds of systems let participants define rules are not part of the TTSM itself and are out of scope of this work. Therefore, the TTSM exposes HTTP REST endpoints that allow the registration of such systems in a loosely coupled fashion. The endpoints supported by this prototype are listed in table 4.4.

Method	**Path**	**Description**
POST	/rules	Registers a new rule service with a callback URL.
GET	/rules	Returns all registered rule services.
GET	/rules/{id}	Returns the rule service with the given ID.
PUT	/rules/{id}	Updates and changes details of a particular rule service.
DELETE	/rules/{id}	Removes a particular rule service.

Table 4.4: List of rules endpoints

Rule validation engines register themselves at the TTSM with a callback URL invoked later on by the *RulesService* to validate commands. The endpoints that rule validation engines must implement in order to be able to interact with the prototype properly are listed in table 4.5.

Method	**Path**	**Description**
POST	/check-new-workflow	Verifies, if the given workflow can be created.
POST	/check-new-instance	Verifies, if the given instance can be launched.
POST	/check-state-transition	Verifies, if the given state transition is allowed.

Table 4.5: List of required rule validation engine endpoints

As described in sections 4.2.5 and 4.2.7, there are two sources of commands that the *RulesService* listens to and sends to all rule validation engines: (1) local commands, that have not yet been transmitted over the network, and (2) the commands received from other participants. In both cases, the *RulesService* subscribes to the event bus of the persistence module in order to properly validate new workflow definitions, instantiations, and state transitions. The command, regardless of its source, is then sent to all rule validation engines registered. Afterwards, the *RulesService* collects all responses and computes the result of the validation and verification process by algorithm 4.2 from section 4.2.5. If the command has been dispatched locally and the result of the validation process is negative, it never enters the network in the first place to prevent blockchain cluttering and to keep a small footprint (see section 2.3.5). If another participant dispatches a command and the validation process is negative, the rules module creates

[33]For example, a facility maintenance contractor has to complete maintenance on an elevator and transmit a document that lists all maintenance steps performed in a predefined structure and order, before the building administrator can perform an inspection. This requires semantic correctness of process execution and the rule validation engines to verify that all required documents are present in the correct form as payloads of previous state transitions.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

a rejection event. A list of events that are dispatched depending on the results of the validation process is given below:

- **Rules.Instance.LocalTransitionAccepted**: A local state transition has been accepted by all rule validation engines.
- **Rules.Instance.LocalTransitionRejected**: At least one rule validation engine has rejected a local state transition.
- **Rules.Instance.ReceivedTransitionAccepted**: All local rule validation engines have accepted a state transition proposed by another participant.
- **Rules.Instance.ReceivedTransitionRejected**: At least one local rule validation engine has rejected a state transition proposed by another participant.

Similar events are dispatched for workflow definitions and workflow instantiations. Note that, as described in the concept in section 4.2.7, participants that dispatch a command (like a state transition, for example) do have to validate it twice, once as a local command and the second time after receiving it through the network. This is because all commands are dispatched to all involved participants. This includes the sender of the command as well. Figure 4.9 illustrates the decision-making process of the rules module on which event has to be dispatched as a response.

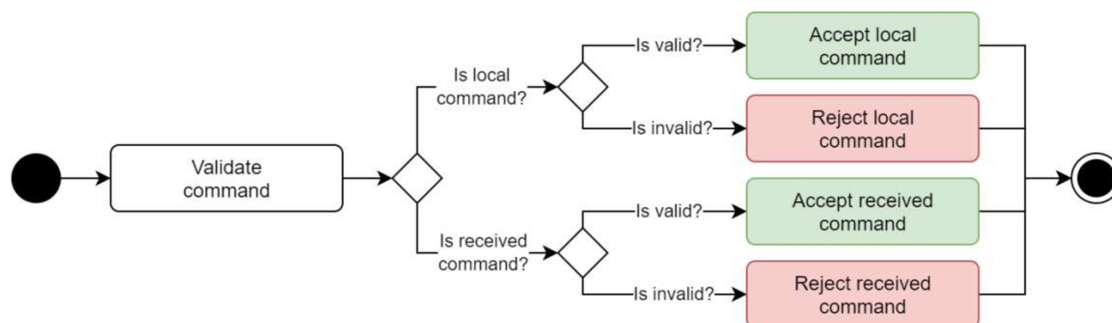

Figure 4.9: UML flowchart diagram of the rules modules decision-making process.

Since the prototype implements the communication with rule validation engines using HTTP, unexpected network timeouts might occur. In this case, the prototype immediately rejects the command, regardless of the response of other rule validation engines, to prevent the command from being accepted that would otherwise have been rejected. This is a design decision made in the prototype since the TTSM concept explicitly delegates the handling of network errors to the implementation (see section 4.2.8). The dispatched events are then further processed by the consistency module.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

4.3.4 Consistency Module

The consistency module, responsible for exchanging messages between participants, is implemented in this prototype using two layers: (1) the application layer being responsible for determining if a message has to be exchanged, and if so, which consistency strategy to use, and (2) the data layer that directly connects to the *EventStoreDB* to listen for events triggering message exchanges. Figure 4.10 depicts the architecture of the consistency module.

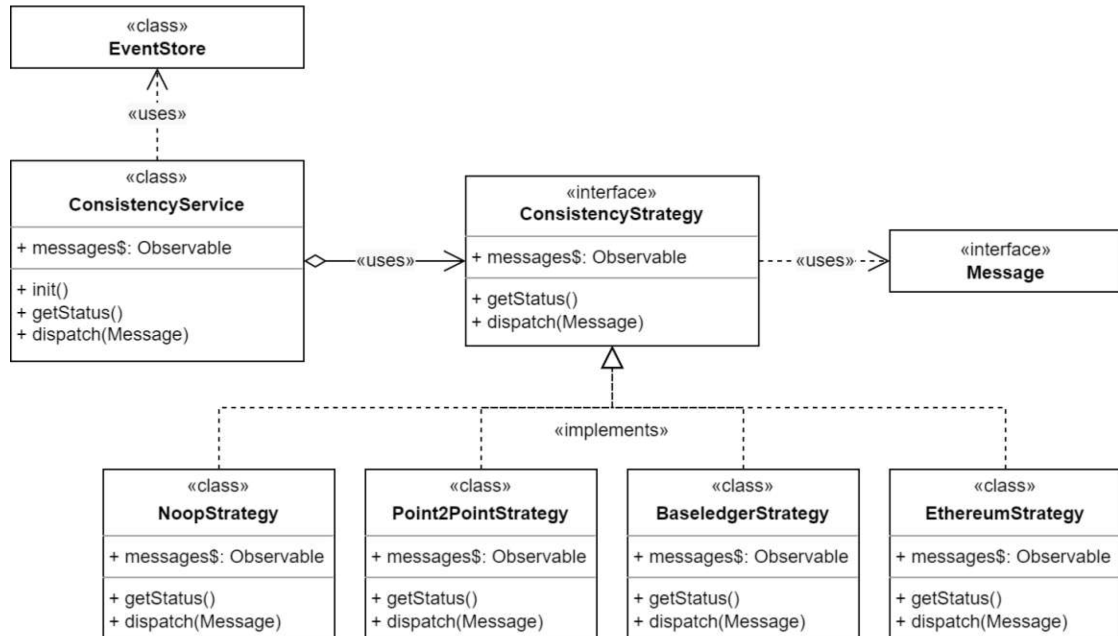

Figure 4.10: Consistency module architecture that enables multi-chain support.

As described in the proposed concept in section 4.2.6, the consistency module receives approval events from the rules module through the persistence module. Depending on the persistence event received, an applicable consistency message is generated. In this prototypical implementation, this is accomplished by the *ConsistencyService*, which subscribes to the event bus. In a fully-fledged TTSM implementation, the *ConsistencySerivce* would now pick an appropriate *ConsistencyStrategy* depending on implementation-specific metrics such as payload structure or consistency message type. However, to keep the complexity of this prototype design at a reasonable scale, the *ConsistencyStrategy* that is being used is predefined on start-up by the developer. This approach might even be sufficient for most TTSM implementations.

Consistency strategies require at least two functions and one field to be implemented. Each consistency strategy implementation has to provide the *messages$* field that implements the observer pattern and emits a new message for each message sent to other participants. It, therefore, represents a constant and congruent stream in each participant's TTSM. The *getStatus* function that returns the status of all required external services (in the

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

case of Ethereum, for example, the status of the node used to write to the blockchain), and the *dispatch(Message)* function that distributes the given message to all participants involved. Some strategies are rather specific, like the *NoopStrategy*, which immediately feeds back the message to the sender (i.e., it can only be used for consistency messages that involve exactly one participant), or the *Point2PointStrategy* that uses HTTP to exchange messages with other participants. A potential use case for the latter one might be the distribution of messages between participants that are not in a conflict of interest, for example. However, most of the communication relies on BCT-based strategies. How these strategies are implemented and what properties these have to fulfill is out of scope of this work. Nonetheless, they are obligated by the TTSM to exchange dispatched messages with other workflow participants reliably and to provide a consistent view of workflows throughout all participants. In this prototypical implementation, BCT-based strategies are primarily implemented in a rather simplistic way.

- The ***EvmStrategy*** generates a smart contract that only stores hashes of the exchanged messages and returns the transaction address as commitment reference. The messages themselves are exchanged using an HTTP-based REST API. It is forwarded to other participants only if the hash was written successfully to the smart contract and, depending on the execution mode, if the commitment reference was attached.
- The ***BaseledgerStrategy*** works in a pretty similar fashion; however, most of the implementation of this strategy is part of the external *baseledger proxy*[34]. Baseledger implements the Baseline Protocol. As described in section 2.5, baseledger uses an internal message bus (in this case NATS[35], a message bus for the edge) to exchange messages between participants. A ZKP is generated as a commitment reference and written to the blockchain, proving that the message was dispatched and distributed correctly.

Once the command that was converted to a consistency message has been distributed to all involved participants using the chosen consistency strategy, the *ConsistencyService* receives the dispatched message (this includes the *ConsistencyService* of the sender as well - no distinction between sender and receiver has to be made from here on out). The message is then converted from a consistency message transmitted over the network and exchanged between participants to an internal persistence event. Afterwards, the persistence event (and the associated command) is evaluated and validated by the rules and the workflow module for syntactic, semantic, and workflow-specific pragmatic correctness as mentioned in the proposed concept in section 4.2.7. As soon as all involved modules have given their feedback, the *ConsistencyService* dispatches another event into the network that indicates that this TTSM either accepts or rejects the command. All participants must wait until N responses were dispatched into the network from N involved participants. If one of these

[34] `https://github.com/Baseledger/baseledger-proxy` (accessed on 2022-09-07)
[35] `https://nats.io/` (accessed on 2022-09-07)

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

N participants rejects the command, it is immediately rolled back. Figure 4.11 shows the simplified network communication between two participants exchanging messages using the TTSM prototype.

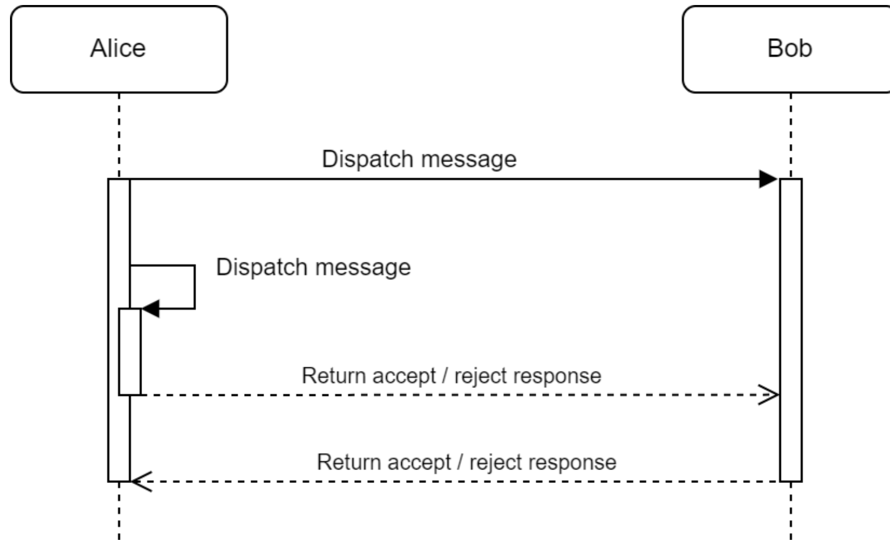

Figure 4.11: Sequence diagram of two participants exchanging messages using the consistency module.

The consistency messages required to perform state transitions are listed below[36]. Each message, as shown in figure 4.4 in section 4.2.6 consists of a message header and an arbitrary payload that is extended by message and command specific metadata.

- **Perform Transition**: Some participant wants to perform a state transition on a given workflow instance. The metadata consists of the name of the state transition (unique identifier), the starting state, and the resulting state.
- **Accept Transition**: Response message of participants if they accept the state transition. Contains the same metadata as the *perform transition* message.
- **Reject Transition**: Response message of participants if their workflow or rules module produced an error while evaluating the state transition. Contains the same metadata as the *perform transition* message.

If an accept or reject message is received, a follow-up persistence event is dispatched by the *ConsistencyService* to allow an examination of participants who rejected and accepted the command. After all responses have been collected, a completion persistence event is emitted as described in algorithm 4.3 in section 4.2.8. It is used to allow the

[36]Similar messages are exchanged for new workflow definitions or instantiations.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Wien Bibliothek Your knowledge hub

consistency module to itself determine when consensus between participants is reached. The persistence events dispatched by the *ConsistencyService* are given below[37]:

- **Consistency.Instance.TransitionReceived**: Created from *perform transition* consistency messages and only dispatched if the workflow instance on which the transition should be performed upon and its workflow definition exist locally.
- **Consistency.Instance.TransitionAcceptedByParticipant**: Created from *accept transition* consistency messages if any of the involved participants approve the state transition.
- **Consistency.Instance.TransitionRejectedByParticipant**: Created from *Reject Transition* consistency messages if any of the involved participants could not perform the state transition due to some issues.
- **Consistency.Instance.TransitionAccepted**: Dispatched by the *ConsistencyService*, if for all N participants involved in the state transition, a *Consistency.Instance.TransitionAcceptedByParticipant* persistence event has been dispatched.
- **Consistency.Instance.TransitionRejected**: Dispatched by the *ConsistencyService*, if for any of the N participants involved in the state transition, a *Consistency.Instance.TransitionRejectedByParticipant* persistence event has been dispatched.

To dispatch persistence events, the *ConsistencyService* connects to *EventStoreDB* in the same fashion as the workflow and rules modules do. This includes creating projections (if necessary) and writing workflow definition events to the *workflows.{id}* stream and workflow instance events, like state transitions, to the *instances.{id}* stream.

4.4 Intrinsic Properties

The knowledge accumulated in the background section 2, together with the research conducted to extract related work in section 3, led to a new concept for workflow execution that leverages on BCTs for verifiability and traceability. This concept, as described in detail in section 4.2, and its prototypical implementation described in section 4.3 gave a sophisticated overview of properties that TTSMs that allow time-travel verification of executed workflows need to fulfill to be viable. The following lists five distinct properties derived from the concept and the implemented prototype and describes them in more detail.

[37] Similar persistence events are dispatched for new workflow definitions or instantiations proposed by other participants.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

Consistent

The *consistency property* guarantees that all participants *eventually* reach consensus and work with the exact same state for all workflows being defined or executed. As described at the beginning of section 4.2, this property is partially enabled by the involved BCTs because their consensus protocols must guarantee at least agreement, integrity, and termination when exchanging messages and advancing the state of a blockchain (see section 2.1.5). The TTSM-consensus algorithm described in section 4.2.8 shows that the TTSM itself can also reach consensus between all involved participants, leveraging on the aforementioned properties that BCTs and their consensus protocols bring to the table. However, due to the increased block time of certain blockchains, consistency can only be archived in the form of *eventual consistency*. Nonetheless, TTSMs that have reached finality can never be inconsistent. If participants, for example, request invalid state transitions, they are rejected immediately, which forces everyone involved to roll back. A similar scenario occurs if two or more participants aim to perform the same state transition simultaneously. Because these participants already locally advanced their state, the same transition cannot be performed twice. Even though the state transition is technically idempotent, the TTSM still rolls back due to potentially diverging payloads. This is ensured by the workflow logic, the persistence module, and the BCT itself.

Persistent

All changes performed within a TTSM and messages exchanged between participants must be available in the form of an event history. This is referred to as the *persistency property*. Persistence in the context of a TTSM is two-fold: (1) the event bus and the event store of each participant are responsible for persisting all exchanged messages and internal TTSM events. Since hostile participants could technically modify this data in their favor, (2) the BCTs also store certain proofs of messages that have been exchanged. Even if a hostile participant tampers their data, the other participants still have undeniable proof that specific actions have been performed. This splits persistence into two distinct areas of concern:

- **Tamper-proofness**: Hinders hostile participants to change workflow execution in their favor in hindsight and generates undeniable proof (see section 4.2.6).
- **Storage**: Enables participants to execute large workflows with varying payloads on public blockchains without exposing privacy critical information (see section 4.2.3) and reducing the footprint on the blockchain, which reduces overall cost (see section 2.3.5).

Both properties are tightly coupled in a TTSM to give participants access to persistence proofs on the blockchain. Due to the employed design principles of the persistence and consistency modules (such as CQRS, event sourcing, or causal ordering), invalid commands can never become a part of the eventually consistent state of a workflow or

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

even the entire system. Such commands are stored in the event history for traceability reasons, but are never incorporated into the state.

Verifiable

Whenever multiple participants in a conflict of interest are involved in the execution of a workflow, the *verifiability property* is of utmost importance because it gives proof that certain events have happened in a particular order at a certain point in time. This property is tightly coupled with the aforementioned *consistency property* 4.4 and the *persistency property* 4.4. Even though hostile participants might modify their local state, the global state, which is stored to some extent on the blockchain, cannot be tampered with[38]. If participants want to prove that some action happened in the past, they can go back in time using the time-travelling capabilities of the TTSM and access the proof on the blockchain associated with the events. Even though not all events might be incorporated into the current system state, they are still part of the event history and hence something participants can verify. Therefore, verifiability is once again a property backed by the underlying BCT and leveraged upon in the proposed concept of a TTSM. The consistency property ensures that the state is consistent in the first place, which guarantees that verification produces the same results for all participants, while the persistency property enables time travel which allows the verification of past states.

Declarative

In order to keep development cost in case of changing requirements at a minimum, and to allow management-oriented users to define new workflows, a TTSM has to fulfill the *declarative property*. Being declarative means that users of a TTSM can entirely configure the execution of workflows without the need for additional software development effort[39]. Being configurable and purely declarative not only enables software architects to use TTSMs as underlying sources of truth for workflow execution, but it also simplifies the integration into existing systems. As described in sections 4.2.1 and 4.3.1, fully-fledged TTSMs might not only support statecharts as the language of workflow modeling but also BPMN, choreography or even process diagrams. Supporting well-established modeling languages further improves the practical value of such a system.

Optimized

Due to highly fluctuating transaction fees of blockchains, the TTSM should reduce the number of messages exchanged to also reduce overall cost [PG17]. This is referred to as the *optimization property* and is enabled by the optimizer described in section 4.2.1 of the proposed concept. Optimizing workflow statecharts can change the syntactic meaning of workflows; however, they must preserve semantics.

[38] Given, that the underlying BCT has enough participants and implements a proper consensus protocol [Bro19].

[39] An example of such a configuration is given in listing 4.1 in the prototype design section 4.3.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

4.4.1 Research Question 2

To answer the second research question:

> *Which properties do BCT-based state machines require to allow time-travel verification of business processes?*

As mentioned above, blockchains already support a vast majority of properties required in BCT-based state machines that allow time-travel verification. However, most related work (as described in section 3.2) leverages on these properties by directly executing workflows on the blockchain. Even though tamper-proofness and consistency are assured, these approaches suffer from a lack of privacy-preserving mechanisms, longer transaction times, a lack of blockchain-acquainted developers, or restrictions to the payload size[40]. Therefore, the set of properties defined for the concept proposed in this work tries to leverage on characteristics of BCTs without creating tight coupling between the blockchain and the workflow execution engine. Properties that TTSMs **must** fulfill in order to be fully functional and to allow time-travel verification of business processes are: (1) the *consistency property*, that ensures that eventually, all participants read the same state of a workflow execution, (2) the *persistency property*, that stores the state in an undeniable and tamper-proof fashion without exposing privacy critical information to uninvolved participants and (3) the *verifiability property*, that enables traceability through time-travel. Properties that **should** be fulfilled in order to make the concept suitable for real-world scenarios are the (1) *optimization property* and (2) the *declarative property*.

Nonetheless, the question remains on how the state of the art must be adapted to fulfill these properties and if integrating a TTSM in existing systems is viable. Therefore, the upcoming sections evaluate the concept by using analytical and architectural analysis and by investigating its utility in simplified real-world scenarios.

[40]Primarily due to native block size limits.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Wien Bibliothek Your knowledge hub

CHAPTER 5

Evaluation

This chapter aims to thoroughly evaluate the practical feasibility of the proposed concept described in chapter 4. To do so, it obeys design science guidelines and, thus, uses methodologies already available in the knowledge base of workflow execution on the blockchain [HMPR04]. This ensures better reproducibility of the presented results and allows in-depth comparison to other approaches. Two artifacts are evaluated in the following: (1) the proposed concept of a TTSM as described in section 4.2 and (2) the instantiation of the concept in the form of the prototype described in section 4.3. Qualitative and static analysis methodologies are applied to the former, while the latter is used for experimental analysis to show real-world utility of the proposed concept.

5.1 Qualitative Analysis

Extracting criteria for qualitative analysis is a rather tricky task, given the heterogeneous landscape of workflow execution approaches that target the blockchain (see related work chapter 3). However, the most often used criteria in related literature can be categorized in flexibility and privacy or security criteria. The following list of criteria was extracted from related literature to allow comparison between different approaches. To name a few, this includes [Pry16, PSHW20, CFR18, SSSJ19, LFW20, WXR^{+}16].

5.1.1 Flexibility Criteria

In BCT-based software, flexibility is a rare trait due to the cost associated with change on the blockchain [Nak09, But22]. Therefore, qualitative evaluation is performed using the upcoming flexibility criteria by creating classes ranging from low flexibility to high flexibility to put the proposed concept into perspective. Notice that higher flexibility typically comes with increased architectural complexity.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

Blockchain Selection

For many more simple real-world scenarios, predetermining a specific blockchain to be used for the entirety of the system or particular workflow instance should be enough. However, for larger and more complex workflows, there is no singular blockchain that fulfills all necessary requirements [LFW20]. Taking this into consideration, and the fact that BCTs are a rather quickly evolving sector, being able to adapt different blockchains becomes a significant point of concern for BCT-based software. Figure 5.1 shows a classification of this particular criteria.

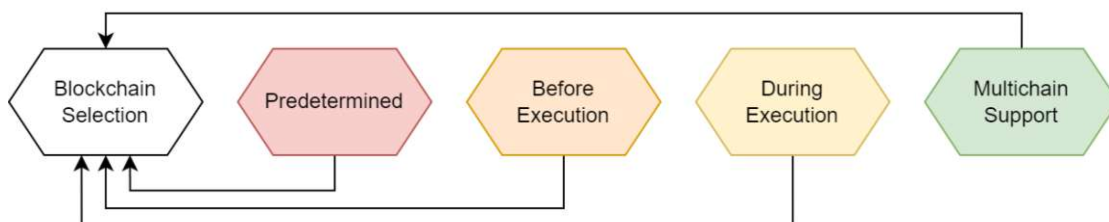

Figure 5.1: A classification for blockchain selection.

Even though there are concepts that enable multi-chain workflow execution[1] like the one proposed by Ladleif et al. [LFW20], most approaches rely on a single blockchain to not only execute workflows but also to tightly integrate the workflow execution system itself with the it [WXR+16, PSHW20, KJ21]. The proposed concept, however, enables participants to select their preferred blockchain, and the associated desired properties, either before or dynamically during workflow execution depending on the workflow itself and the messages exchanged. This increases flexibility but also complexity due to the abstraction required.

Participant Selection

Short-running business processes and their corresponding workflows might rely on a fixed set of participants without threatening the stability or flexibility of a system or workflow. However, more volatile workflows with a considerably larger amount of participants involved need a more flexible system that enables the executing participants to select who executes certain activities on demand [FRMR12]. Furthermore, it is advantageous for the overall workflow to select participants only after the workflow has already been launched. This allows executing participants, especially in longer-running workflows, to take environmental and political change into consideration when selecting a contractor or supplier that is required later in the workflow, for example. The classification for participant selection depicted in figure 5.2 differentiates between three classes ranging from least to most flexibility.

[1] i.e., using multiple blockchains at once that are interconnected with each other and can even exchange messages.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

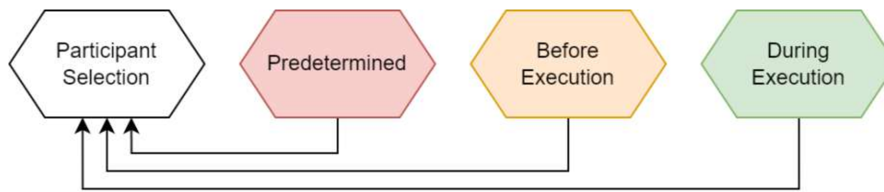

Figure 5.2: A classification for participant selection.

Systems that predetermine all participants that are involved in the execution of workflows have not been found in related literature; however, the number of approaches that require participants to be determined before the workflow is instantiated (e.g., in [LWW19, NMK18]), and the number of approaches that allow dynamic, or even on-demand participant selection during execution (e.g., in [LSNW20, PSHW20]), is about even. However, the concept proposed in this work does not explicitly state how participant selection should be implemented. Even though workflow definitions must specify the number of parties involved and which role they play in the overall choreography, choosing a tangible TTSM of a participant that has been selected at a later point in time during workflow execution is still viable. This is because roles are only loosely coupled to the actual participants that carry them out. However, actions performed by participants that are no longer part of a workflow execution remain on the blockchain. This is a trait required by the persistency and verifiability properties of a TTSM that might contradict privacy concerns to some extent[2]. Nonetheless, the proposed concept technically allows any participant selection mechanism to be implemented.

Workflow Mutability

Being able to update workflows during execution increases flexibility even further; however, it also creates additional complexity in the form of potential issues regarding (eventual) consistency. Figure 5.3 shows a potential classification for workflow mutability.

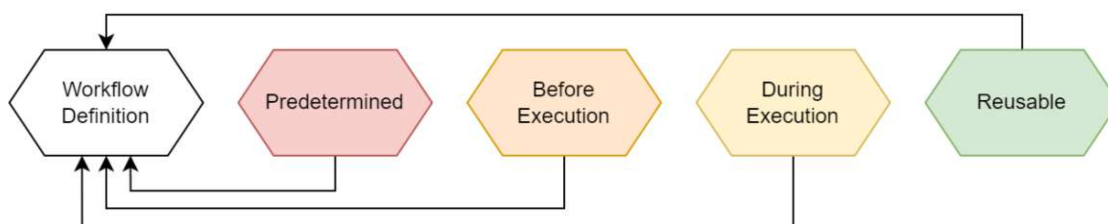

Figure 5.3: A classification for workflow mutability.

Besides predetermining workflows as part of the system being the most inflexible solution, allowing participants to define workflows themselves before execution and instantiation

[2]Given that no longer involved participants request data that has been associated with their participation, to be entirely removed according to the European GDPR for example.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

of the workflow itself, is the most common approach [WXR+16, KJ21, SSSJ19]. The proposed concept of this work also falls in this category performing a balancing act between complexity and reusability. Even though technically possible to adapt, allowing workflow mutability during execution or even partial reusability of already executed workflows (as described in [LSNW20], for example) requires some changes to the concept itself. In this context, open questions like eventual consistency and participants working on different states of a workflow caused by an increased transaction inclusion duration[3] on certain blockchains or how to adapt reusability in BP-centric workflow execution engines remain yet to be answered.

Modelling Language

Support for a wider range of modelling languages for workflow definitions not only improves flexibility, but also readability and reusability, because already existing process models in organizations can directly be fed into the system without the need of further adaptation. This prevents possible errors from being introduced while manually converting said models. Figure 5.4 includes two classifications for modelling language flexibility.

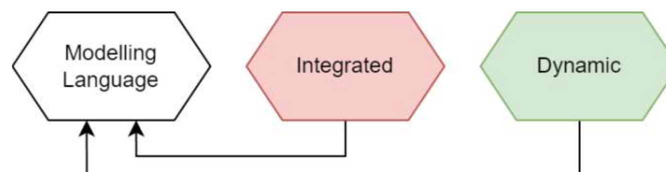

Figure 5.4: A classification for modelling language support.

While systems, where the modelling language is tightly integrated, are strictly bound to the language semantics, dynamic approaches might require translation from one language to another. The dedicated conversion step in the workflow module (see section 4.2.1), puts the proposed concept on the most dynamic end of the spectrum compared to related work. Due to the internal usage of statecharts, all process models, that can be reduced to statecharts, can technically be supported. Transformations to statecharts, however, are out of scope of this work due to their complexity. Nonetheless, algorithms have been published in recent years and are publicly available for possible adaptation in a TTSM [NMK18, ZHJ04].

5.1.2 Privacy and Security Criteria

Privacy is an often neglected criterion regarding workflow execution on the blockchain. In most related work, authors advise the usage of private blockchains to counteract this problem [WXR+16, KJ21, RCDF20, SSSJ19, LSNW20, LBAG21]. However, this approach only partially solves the issue, and some open problems remain. Companies might want to treat parts of internal workflows as a trade secret because they are of

[3] A commitment reference is not yet available and the command therefore not multicasted.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

utmost importance to the business's success or because they are simply not relevant to the shared BP. Another example is confidential information that is only allowed to be shared between two particular participants and must never be exposed to others[4]. Furthermore, the usage of private blockchains can threaten security, as previously discussed in chapter 3, due to the increased voting power per participant compared to public blockchains. Similar to the flexibility criteria, the upcoming privacy and security criteria are also divided into classes to put the proposed concept into perspective.

Workflow Structure Sharing

Workflows that involve multiple independent participants do require sharing of workflow structures in some form to create a common frame of reference. Figure 5.5 differentiates between three public and one private classification. First, public workflows that share not only their structure but also their execution context[5]. Second, public actions, where only state transitions are exposed; and third, public metadata, where only (encrypted) metadata about the workflow is shared publicly. Purely private workflows are the fourth class, in which no information is put on a blockchain that is used by more than one participant.

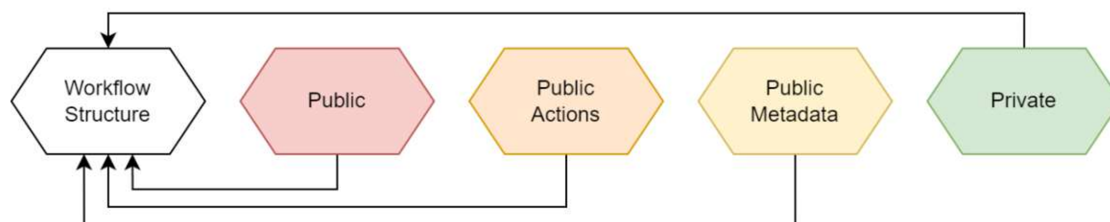

Figure 5.5: A classification for sharing workflow structures.

Depending on the implementation of the consistency module of a TTSM and the chosen consistency strategy, the proposed concept might either fall into the second or third category — sharing actions or metadata on the public blockchain, but is never executing entirely on-chain (restricted by the architecture by separating the workflow module from the consistency and persistence modules). Strategies for sharing workflow structures might include transmitting actions off-chain but storing hashes or ZKPs of actions being performed on-chain. Furthermore, the proposed concept aims to only share choreography structures that determine the required interactions between participants. Internal workflow structures can be kept entirely private or semi-private by leveraging on homomorphic encryption or ZKPs, for example.

[4]Imagine a four-party scenario where three banks and the government are involved. All participants agree on using a blockchain as a trusted third party. Some information about customers, however, is only allowed to be shared with the governmental instance and not with any other participant due to privacy regulations. Private blockchains cannot solve this issue if the privacy critical data is stored on-chain because all four participants can investigate the state of the blockchain at any point in time.

[5]In the form of smart contracts, for example.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

Data Sharing

The way workflow execution engines share data associated with workflows and state transitions is categorized by the classification given in figure 5.6. It differentiates between data being shared over a public network and thus being available to all workflow participants, only sharing (encrypted) metadata or keeping data entirely private.

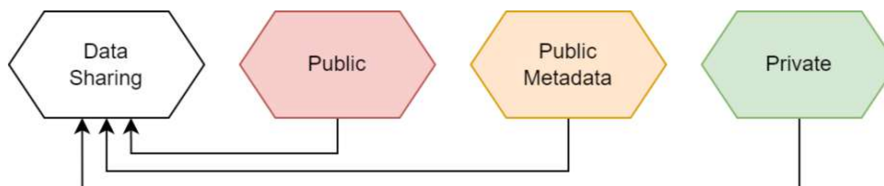

Figure 5.6: A classification for sharing data between participants.

Most approaches found in related literature either require participants to share data entirely off-chain or entirely on-chain. The former approach neglects traceability capabilities of BCTs, and the latter restricts payload size to the maximum allowed block and transaction size of the (most often) predetermined blockchains being used. Other approaches, like the ones proposed by Lichtenstein et al. [LSNW20], and Ladleif et al. [LWW19], which are laying their focus on the artifacts produced by workflows, also store the state and life cycle of the entire artifact on-chain and thus, makes them publicly (in the sense of “for all participants”) available. However, the concept proposed in this work aims to bridge the gap in the related literature by only storing metadata on-chain. Similar to the workflow structures described in the previous criteria 5.1.2, a TTSM only stores hashes of actions and their associated payload data or ZKPs, for example, that ensure, that the data fulfills specific properties. Open questions, especially regarding techniques like ZKPs or homomorphic encryption, must still be solved in future work to fully leverage on the capabilities of the proposed concept.

Trust

Trust between participants is vital when they are in a conflict of interest. Studies have shown that incorporating trust as a key ingredient into workflow systems might increase workflow performance, particularly in supply chain management and logistics [FVB05, JMSK04]. A common solution to establish trust is to utilize trusted third parties [PL09, AGI$^+$19]. All approaches discussed in chapter 3 build upon the blockchain to take up this particular role. So does the concept proposed in this work. To enable traceability through the blockchain, the consistency module of a TTSM associates exchanged messages with commitment references. This ensures that all actions that require multi-party participation are undeniably persisted and can be verified. However, the proposed concept does not support authentication mechanisms. This means that participants occupying a particular workflow role might not be who they pretend they are. Even though workflow execution does not inflict trust-related issues, the open authentication problem might. To participate in a workflow that is handled by a BCT-based TTSM,

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

participants require a blockchain wallet in some form to at least provide a pseudo-identity[6]. However, more is needed to authenticate participants trustfully because a wallet might also belong to an attacker pretending to be a particular participant. Thus, authentication remains an open problem which is out of scope of this work due to its complexity. Future work might consider the usage of asymmetric cryptography and signatures in order to solve this issue.

Conflict Resolution

Resolving conflicts between participants using undeniable proofs is one of the cornerstones that blockchains provide. The proposed concept leverages on this property by referencing commitments on the blockchain for each action performed by any of the participants. Therefore, conflicts can be resolved by the time-travel capabilities of a TTSM - participants involved in a conflict can travel back in time to the point where the conflict arose. The workflow at this point in time contains all actions that led to the current state and, thus, all associated commitment references on the blockchain, which in fact, are undeniable proof of what happened[7]. Nonetheless, a follow-up issue arises regarding the authentication of participants mentioned before in criteria 5.1.2. Even though a role might have performed an action according to the blockchain, this does not give proof that the action has indeed been performed by the correct participant. Due to the pseudo-anonymity of blockchains, one can not easily prove the identity of contributors without further usage of off-chain techniques such as legal contracts associating a participant to a wallet address, for example. Resolving the authentication and identity issue in future work might also solve the conflict resolution issue mentioned in this section.

5.1.3 Summary and Discussion

The qualitative analysis performed in this section has shown significant traits of the proposed concept regarding flexibility compared to existing solutions. For the most part, this can be traced back to the role of the blockchain that is only of supportive nature, a characteristic that has not been investigated in related work and BCT-based workflow execution yet. By leveraging on the blockchain without statically integrating it as a software architectural cornerstone, TTSMs are not bound to restrictions given by the blockchain ecosystem. Compared to other concepts, this allows the concept proposed in this work to be more flexible and extensible, resulting in a highly modular architecture where components are only loosely coupled and interchangeable at any point in time. This kind of flexibility has been an unseen trait in BCT-based workflow execution engines and results in a list of advantages that are otherwise difficult to obtain, including TTSMs switching blockchains during workflow execution without significant

[6]BCTs provide pseudo-anonymity, thus, it is practically almost impossible to authenticate a participant by her wallet information.

[7]Given that the blockchain itself is tamper-proof (see consensus section 2.1 for more details).

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

complexity overhead[8], dynamically selecting participants on-demand, or changing the (shared or internal) workflow definition during runtime. However, two gaps in the state of the art and the proposed concept have been identified regarding blockchain interaction and authentication of participants:

- Investigation of applicable techniques that store minimum amounts of data on-chain but provide maximum traceability and verifiability.
- Investigation of authentication techniques for blockchain participants, to prove identity (also referred to as decentralized identifiers[9]), to seal attack vectors like Sybil attacks.

Solving these two research problems (1) further reduces transaction cost while not endangering the traceability and verifiability that TTSMs already provide, and (2) being able to undeniably associate actions being performed by participants with real-world identities. Nonetheless, the qualitative analysis has shown that the proposed concept combines traits from different approaches while reducing the footprint on the blockchain, preserving privacy to some extent, and improving flexibility.

5.2 Static Analysis

In this section, the static structure of the proposed concept is analyzed by deriving formal metrics and taking a closer look at the macro software architecture. It aims to give insight into the complexity and cost of a TTSM. In order to do so, the remainder of this section assumes the usage of a layer-1 BCT-based consistency strategy in the consistency module to assure comparability with related work [WXR+16, LBAG21, PSHW20, LPDGBW19]. The BCT-based consistency strategy creates a peer-to-peer network where each participant is directly connected to all the other participants. An additional connection to the blockchain is established that is used to prove the integrity of each message exchanged. Other technologies, such as layer-2 rollups or using ZKPs, might yield other results.

5.2.1 Network Topology

The TTSM network topology is determined by the number of workflow participants, only including the ones that are relevant to perform a specific command. Even if the workflow requires a total of N roles, certain state transitions might only interact with $n < N$ participants occupying a subset of the N roles. In the following, state transitions are used exemplary for any kind of command w.l.o.g. Given a state transition that

[8] Note that, regarding practical feasibility, participants have to determine which blockchain is best suited for which message types in order to guarantee flawless transitions during execution. Furthermore, data remains on the blockchains it was initially written to. Time travel and verification, therefore, also includes switching between blockchains.

[9] https://w3.org/TR/did-core/ (accessed on 2022-11-29)

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

requires interaction with n participants, the *maximum* number of connections each participant has to establish can be derived from the fact that the TTSM concept operates on a peer-to-peer network. Thus $n - 1$ connections must be established, excluding the participant that proposed the state transition. Therefore, the total number of connections required is equal to the number of edges in a complete graph. The computation is given in equation 5.1 with C_{max} being the total number of connections required.

$$C_{max}(n) = \frac{n(n-1)}{2} \tag{5.1}$$

The proposal of a single state transition only requires the proposing participant to connect to $n - 1$ (excluding self) involved participants. Each involved participant, excluding the proposing one, has to establish precisely one connection resulting in a total amount of $n - 1$. However, after receiving the state transition, each participant has to send an accept or reject message to all other $n - 1$ participants. Thus, resulting in a complete graph or "fully connected mesh". One can now derive the total number of messages exchanged M when performing a state transition[10] using equation 5.2.

$$M(n) = n^2 + n \tag{5.2}$$

The number of messages per state transition is therefore quadratically bound to the number of involved participants assuming that a message is not only sent to the other $n-1$ participants but also to one self[11]. Expressed in Landau notation, this results in an overall communication complexity of $\mathcal{O}(n^2)$ given that $n > 1$ when performing actions w.l.o.g.

5.2.2 Blockchain Transactions

Similar to the network topology, the maximum number of blockchain transactions is also determined by the number of involved participants required for a specific command. Given a total of N roles and n participants, launching a workflow might involve $n \geq N$ participants[12], however, performing state transitions most of the time requires $n \leq N$ participants. Thus, the maximum number of transactions on the blockchain is linear to the number of involved participants, as shown in equation 5.3 where TC is the blockchain *transaction count* and t_p the number of *participants* involved in the command.

[10] Starting in the proposal phase until the transaction has been included into a block.

[11] Imagine an exemplary scenario, where four parties are involved: (1) the product manufacturer, (2) the reseller, (3) the customs authorities and (4) the freight forwarder. When the reseller orders a product, only the reseller, and the manufacturer have to perform and accept the state transition (i.e., all participants involved in an activity). Given that $N = 4$ and $n = 2$, the total number of messages exchanged over the network is $2^2 + 2 = 6$ and the number of connections is $\frac{2(2-1)}{2} = 1$. The customs authorities and the freight forwarder may not receive any messages at all at this point.

[12] A single role might be fulfilled by different participants at different points in time.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

$$TC(t) = \begin{cases} 0 & \text{for } t_p < 1 \\ t_p + 1 & \text{for } t_p \geq 1 \end{cases} \tag{5.3}$$

Commands involving one participant (i.e., themselves) typically do not require blockchain interaction. However, in certain scenarios, a participant might want to prove to an external party that a specific artifact existed at a particular point in time. Given that a TTSM solely operates on statecharts, one can derive the total number of blockchain transactions per workflow instance as follows. Let W be a workflow defined as statechart and formalized as 5-tuple $\langle S, s_0, F, E, T\rangle$ where S is the set of possible states, $s_0 \in S$ the initial state, $F \subseteq S$ being the set of final states and T is the set of transitions. Then, the total amount of required blockchain transactions is linear to the cardinality of T denoted as $|T|$ and, in case of loops or decisions in a workflow, the number of state transitions t_c performed per $t \in T$ with TC_{total} being the *total transaction count* on the blockchain.

$$TC_{total} = \sum_{t \in T} TC(t) \cdot t_c \tag{5.4}$$

Equation 5.4 computes the maximum number of blockchain transactions in a workflow instance as a summation of state transitions $t \in T$ and the associated number of required participants. Leveraging on the notation defined above, one can derive the average number of participants involved per state transitions $\overline{n}$ as follows.

$$\overline{n} = \sum_{t \in T} \frac{t_p}{t_c} \tag{5.5}$$

Given $\overline{n}$ participants that have been required to complete workflow instance I on average and m being the total number of state transitions performed, the overall communication complexity is linear and comes down to $\mathcal{O}(\overline{n} \cdot m)$ which simplifies to $\mathcal{O}(m)$ assuming $\overline{n}$ being the smaller factor.

5.2.3 Persistence Events

The number of persistence events dispatched by a TTSM always has to equal or be greater than the number of blockchain transactions (see section 4.2.7). This is a constraint required by the persistence and verifiability properties described in section 4.4 and can be formally expressed as $TC_{total} \leq EC_{total}$. To compute EC_{total}, one must first compute the number of persistence events per state transition. Let T be the set of possible state transitions in an arbitrary statechart defined as $\langle S, s_0, F, E, T\rangle$. Then, the number of persistence events dispatched per state transition $t \in T$ is computed as follows.

$$EC(t) = (t_p + 1) + (t_r + 1) + 1 \tag{5.6}$$

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek WIEN Your knowledge hub

In the first step, a single participant requests a state transition. This state transition is received and persisted by all involved participants. Afterwards, the local rule system dispatches a persistence event (including the validation results) for each rule engine registered t_r and one that indicates if the overall validation process was successful. In the last step, each participant has to either send an accept or reject message over the network, which results in a single persistence event per participant t_p. An additional event is dispatched to indicate if all involved participants accepted or at least one rejected the proposed state transition. Participants not involved in a state transition do not emit any persistence events. Therefore, the total number of persistence events EC_{total} can be computed as the sum of persistence events per state transition performed.

$$EC_{total} = \sum_{t \in T} EC(t) \cdot t_c \tag{5.7}$$

One observation when comparing EC_{total} to TC_{total} is that EC_{total} is always larger than TC_{total} because, in addition to the events dispatched per participant, $EC(t)$ also includes the events dispatched by the rules module and an additional event that indicates the final status of the command.

5.2.4 Summary and Discussion

The formal static analysis performed in this section has shown that the overall complexity of the network, required in a TTSM, is bound quadratically to the number of participants due to the underlying "fully connected mesh" topology. This increases resiliency because, in theory, participants can forward messages to other participants, but it also increases the complexity of the required infrastructure. Figure 5.7 depicts a possible setup between three participants, namely Alice, Bob, and Mallory, where the direct link between Alice and Bob has been lost, and Mallory must forward messages.

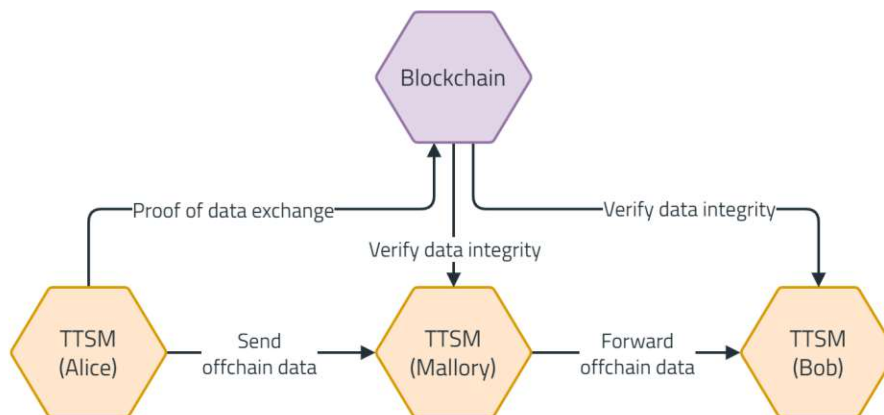

Figure 5.7: Mallory forwards messages from Alice to Bob.

Setups like these are one possible mechanism supported by a TTSM network that allows compensation of direct peer-to-peer links being lost. However, each participant must

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

retain her connection to the blockchain to persist proof of data being exchanged and to verify the integrity of (forwarded) messages. Imagine a scenario where Alice dispatches a state transition request that Mallory must forward to Bob to complete the state transition. Instead of forwarding the original request dispatched by Alice, Mallory modifies it to make Bob accept the message and tamper with the state in a way that favors Mallory slightly and puts Bob at a disadvantage. Even though such attacks might be dangerous because they reduce the overall trust in the network, it is possible for Alice, and especially Bob, to verify the integrity of the forwarded message using the blockchain[13].

To better integrate compensation mechanisms into the TTSM concept, a new consistency message that proves that a participant has indeed forwarded a message could be introduced. Two opportunities for future work have been identified:

- The investigation of compensation mechanisms for lost peer-to-peer or even peer-to-blockchain connections to further increase resiliency.
- The investigation of algorithms with a small on-chain footprint that allow participants to verify message integrity.

Furthermore, this section has shown that the number of blockchain transactions required is linear to the number of participants involved in a particular state transition and that the number of persistence events dispatched internally by a TTSM is strictly larger. Internal persistence events are not only dispatched for each blockchain transaction (associated with a commitment reference), but they can also be used for everything that happens only on the side of a single participant that is irrelevant for other participants. This enables a TTSM to give participants a much finer granularity on traceability without polluting the blockchain.

5.3 Scenario Simulations

The aim of this section is to show the real-world utility of the approach proposed in this work. Different use cases are therefore simulated to conduct an experimental analysis of the concept and its prototypical implementation. Furthermore, with a strong focus on execution cost and latency, the results are compared with approaches from related work. The primary sources, used as baseline references for the upcoming evaluation, include the approaches proposed by Prybila et al. [PSHW20], who conducted a thorough evaluation of the execution duration of their approach, Weber et al. [WXR+16] who gave an overview of the execution cost and latency when applying their concept to a supply chain and incident management use case, and Loukil et al. [LBAG21] who evaluated their approach against two compiled and two interpreted concepts from related literature regarding execution cost.

[13]For example, by comparing the hash of the forwarded message with the hash stored on the blockchain.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

5.3.1 Prototype Adaptations

Even though the high flexibility of the proposed concept, as shown in section 5.1.1, would allow the usage of an arbitrary blockchain or layer-2 rollup, an Ethereum Virtual Machine (EVM)-based approach is used to improve comparability with existing approaches from related literature. Using the EVM furthermore allows Solidity byte code to be converted and ported to other blockchains and rollups such as zkSync[14] or Optimism[15] as well. The EVM consistency strategy implemented for this evaluation creates a fully connected peer-to-peer mesh network where each participant directly communicates with all the other participants. In addition, each participant establishes a connection to the EVM to store the SHA-256 hash of the exchanged message on-chain. The smart contract developed and deployed for this purpose is shown in code listing 5.1.

```
pragma solidity >=0.7.0 <0.9.0;

/**
 * @title Hash Storage
 * @dev Store 256-bit hash values as event log
 */
contract HashStorage {

    /**
     * @dev Stores all hashes in an event log. Using events reduces
        ↪ gas cost dramatically.
     */
    event StoreHash(bytes32 hash);

    /**
     * @dev Stores the given hash values as event log.
     */
    function store(bytes32 hash) public {
        emit StoreHash(hash);
    }
}
```

Listing 5.1: Implementation of a smart contract that stores message hashes

This very simplistic smart contract only allows the storage of 256-bit values, which is exactly the amount of data an SHA-256 hash requires. To further reduce cost, the hashes are stored as event logs only. This enables traceability but omits the need for expensive G_{sset} operations with a gas cost of 20000, only leaving a $G_{transaction}$ operation as the single most expensive operation with 21000 gas in this smart contact. With G_{log} and $G_{logtopic}$ requiring 375 gas, and $G_{logdata}$ only requiring 8 gas per byte stored, each hash persisted on the EVM requires less than 23000 gas [Woo22]. The integrity of an

[14] https://zksync.io/ (accessed on 2022-11-25)
[15] https://optimism.io/ (accessed on 2022-11-25)

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

exchanged message can then be verified by fetching the transaction receipt and testing if the hash stored on the EVM is the same as the hash of the message's payload. This process is part of the EVM strategy and is depicted in code listing 5.2.

```
async receiveConsistencyMessage<T>(msg: ConsistencyMessage<T>) {

  // Retrieve transaction receipt to check if log contains correct
      ↪ message hash.
  const txHash = msg.commitmentReference.transactionHash;
  const tx = await Web3.eth.getTransactionReceipt(txHash);

  // Reject the message if the expected and the actual hash of the
      ↪ message payload differ.
  const expectedHash = tx.logs[0].data;
  const actualHash = sha256(JSON.stringify(msg.payload));
  if (expectedHash !== actualHash) {
    return 'INVALID_HASH';
  }

  // Pass the message on to other modules.
  actions$.next(msg);
  return 'OK';
}
```

Listing 5.2: Implementation of the verification process of messages received

If the hash differs from what has been stored on the EVM, the message is entirely omitted because its integrity cannot be verified. For reproducibility reasons, the simulations rely entirely on the EVM strategy described above. The prototypical implementation used for this evaluation is available GitHub[16]. The upcoming sections introduce the use cases the prototype is evaluated against, derive corresponding BPMN diagrams, and afterwards discuss the results of the simulation runs and compare them to results from related literature.

5.3.2 Scenario Descriptions

Three distinct scenarios have been chosen for the evaluation of the proposed concept. The first one simulates a simplified facility maintenance use case that has been created throughout the course of this work by conducting interviews with domain experts - introduced in section 2.4.1, this scenario aims to show real-world utility of the proposed concept. The second scenario simulates a supply chain, and the third a software incident management use case as described and adapted by Weber et al. [WXR+16], López-Pintado et al. [LPDGBW19], and Loukil et al. [LBAG21] to improve comparability and reproducibility of the results. The second and third scenarios are only briefly outlined and not further discussed.

[16] https://github.com/danielkleebinder/ttsm-prototype (accessed on 2022-11-29)

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Wien Bibliothek Your knowledge hub

Facility Management

The facility maintenance use case, as already partially introduced in section 2.4.1, has been derived from interviews conducted with real-world domain experts. It describes a scenario where a building administrator has been notified[17] that maintenance on a facility inside the building[18] is due. The building administrator now contacts an external maintenance contractor and prepares the facility for further inspections and the maintenance itself[19]. Afterwards, the contractor starts performing the maintenance, orders spare parts if repairs are required, and sends the building administrator a notice that maintenance has been completed. After the administrator's successful inspection of the facility, the maintenance contractor sends an invoice and a maintenance report. The BPMN diagram of this scenario is depicted in figure 5.8.

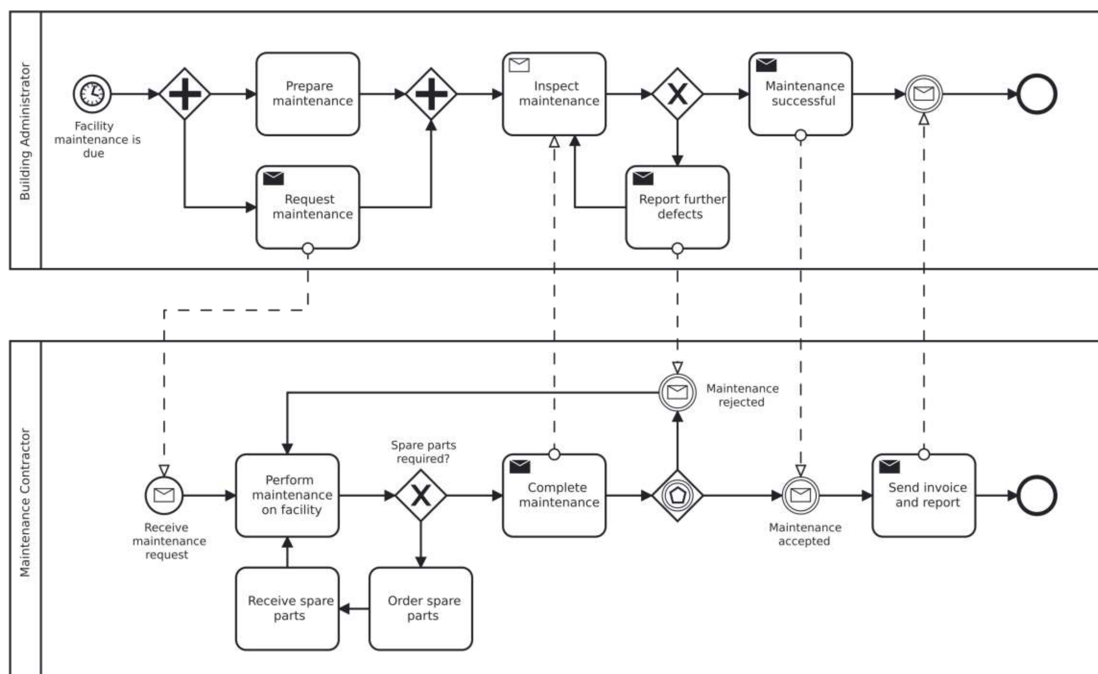

Figure 5.8: BPMN diagram of the facility maintenance use case introduced in 2.4.1.

This use case should demonstrate the real-world utility of the proposed TTSM approach. It has been simplified to some extent to use it as a stepping stone towards more complex scenarios. However, it illustrates the use of parallel and exclusive gates and the exchange of messages between participants. Therefore, the facility management scenario is an excellent first scenario to show that the proposed concept would indeed be functional in real-world environments.

[17]Either through a timely trigger or thorough inspection conducted by a third party, for example.
[18]This might be an elevator, a vending machine, or an escalator, for example
[19]By closing off the facility site, for example.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

Supply Chain

The supply chain use case is commonly used in related work to demonstrate the utility of a new approach and to compare its cost with other approaches. First introduced by Fdhila et al. [FRMKR15], Weber et al. adapted and simplified this scenario in their work [WXR+16]. It consists of five participants interacting with each other. A bulk buyer orders a product from a manufacturer. The manufacturer then calculates what supplies in the form of raw materials and basic resources are required to produce this product and orders them from a middleman. The middleman forwards the order for the required supplies, and a carrier transports them from the supplier to the manufacturer. The manufacturer then produces the requested product and delivers it to the bulk buyer.

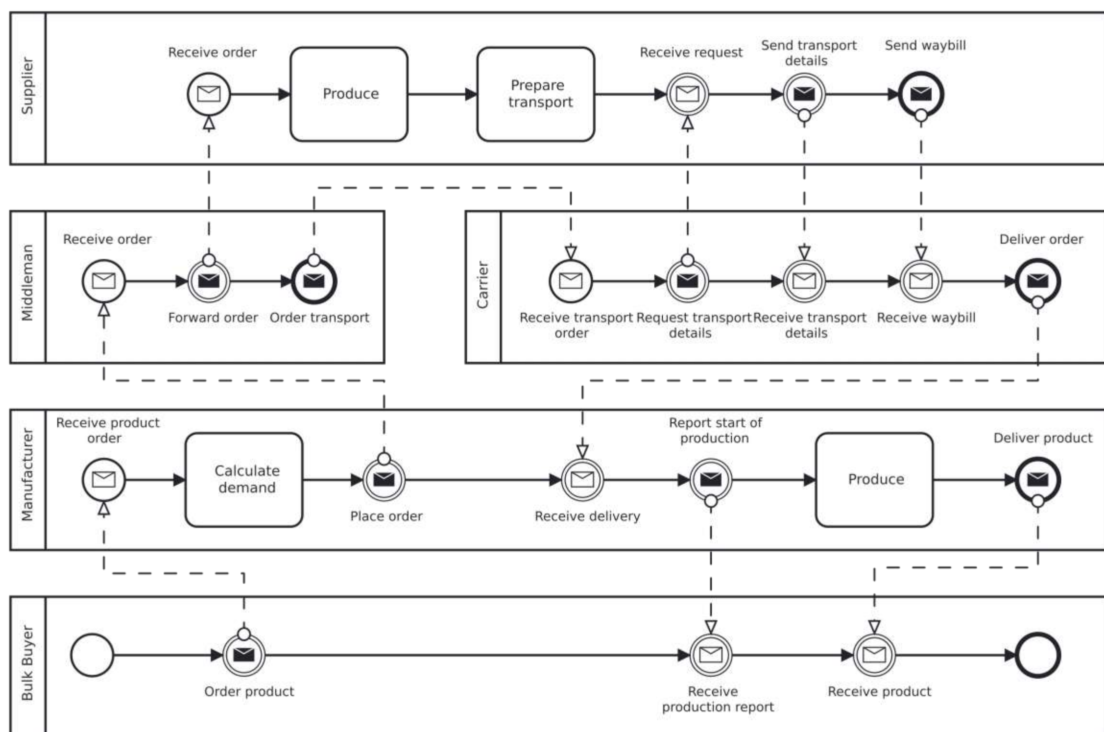

Figure 5.9: BPMN diagram of a supply chain adapted from Weber et al. [WXR+16].

One might realize that this supply chain has much conflict potential. Imagine the bulk buyer ordering a product that must arrive before a deadline. If the deadline is not met by the manufacturer, she has to pay penalties. To be on time, the manufacturer orders supplies through the middleman. However, the requested supplies arrived three days later than expected and were not in the correct quantity. This puts the manufacturer, whose time frame now has shifted, in a tough spot and the carrier since the manufacturer might refuse to accept the incomplete delivery. Therefore, the supply chain scenario can be used quite well to demonstrate conflict resolution capabilities [WXR+16, LBAG21].

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

Incident Management

The incident management use case is, similar to the supply chain use case, a commonly encountered scenario in related work. First discussed in “BPMN 2.0 by Example” [OMG10], it found its way into the domain of workflow execution on the blockchain through Weber et al. [WXR+16] and was then further adapted by many more authors in the sense of design science for comparability reasons [GBPDW17, LPDGBW19, SSSJ19, LBAG21].

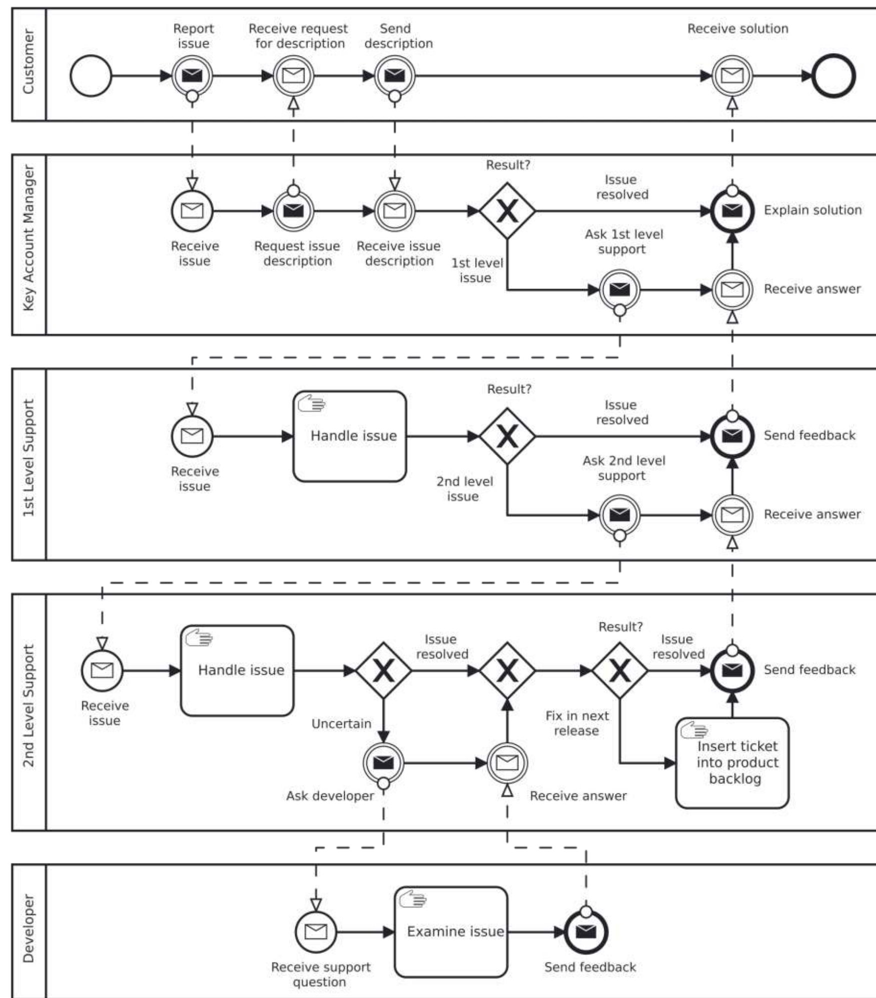

Figure 5.10: BPMN diagram of the incident management use case adapted from “BPMN 2.0 by Example” [OMG10].

Figure 5.10 depicts the BPMN diagram of the entire incident management use case. It follows an issue reported by a customer who noticed a problem with a particular software component. The issue then goes through multiple support layers, starting with the key account manager. If the key account manager cannot solve the issue by herself, she hands it over to the 1st level support. If the 1st level support cannot resolve the

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek WIEN Your knowledge hub

issue, it is passed on to the 2nd level support who either asks a software developer for assistance or immediately creates a ticket and puts it into the products backlog. This relatively sophisticated business process includes more gateways and a strict separation of participants[20].

5.3.3 Discussion of Results

The evaluation results in the upcoming sections have been obtained by extracting only the communication between participants as choreography diagrams. The choreography diagram for the facility management scenario is depicted in figure 2.3. For the supply chain and incident management scenarios, the same choreography diagrams have been used as in the related literature. These diagrams are both depicted in the work of Loukil et al. [LBAG21]. Afterwards, semantically equivalent statecharts have been derived using the Stately editor[21] and fed into the TTSM prototype. Finally, a script was employed to automate the process of deploying, instantiating, and executing the scenarios to ensure that each sample is executed with comparable constraints and conditions. Table 5.1 gives a brief overview of the structure of each scenario.

Scenario	Participants	Tasks	Gateways
Facility Maintenance	2	5	1
Supply Chain	5	10	2
Incident Management	5	9	6

Table 5.1: Structural comparison of the choreography diagrams and statecharts of the evaluated scenarios

The supply chain and incident management scenarios used in this work have the same structural properties as those used for evaluation in related literature [WXR+16, GBPDW17, LPDGBW19, SSSJ19, LBAG21]. This not only improves comparability but also reproducibility of the results. The evaluation itself is also fully automated and can be triggered after setting up the prototype project and specifying an applicable consistency strategy.

Confirmation Correctness

The capabilities of detecting incorrect workflow traces have been evaluated according to the methodologies defined by Weber et al. [WXR+16] and Loukil et al. [LBAG21]. A workflow trace is a path from one task in a workflow instance to another final task. For the evaluation, 500 traces have been randomly generated by shuffling a predefined set of all state transitions T of a given workflow W to generate a new set of distinct traces D, where D has a cardinality of 500. Furthermore, $\forall T' \in D : T' = T$, however, T' has a unique permutation of elements in D. This results in two subsets $D_c \subseteq D$ and $D_n \subseteq D$,

[20]Developers can only talk to 2nd level support, 2nd level support can only talk to developers and 1st level support and so on.

[21]`https://stately.ai/` (accessed on 2022-10-10)

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek WIEN Your knowledge hub

where D_c is the set of conforming traces[22] and D_n the set of non-conforming traces[23] such that $D_c \cap D_n = \emptyset$. All traces have been correctly identified by the TTSM prototype according to their correctness. A list of the results per scenario can be found in table 5.2.

Scenario	Type	Samples	Correctness
Facility Maintenance	Conforming	206	100%
	Non-Conforming	294	100%
Supply Chain	Conforming	180	100%
	Non-Conforming	320	100%
Incident Management	Conforming	197	100%
	Non-Conforming	303	100%

Table 5.2: Average execution duration of each scenario

The results show that the proposed concept performs equally as well as existing solutions regarding the identification of correct and incorrect traces [WXR+16, LBAG21]. Nonetheless, the TTSM prototype has shown better resiliency and a higher tolerance for conforming traces because it can recover when an invalid state transition event has been dispatched. This is caused by the underlying XState library that ignores invalid state transitions and preserves the current state. Overall, the results presented in this section are in line with expectations.

Blockchain imposed Latency

Overall latency of the system and ensuring finality or even just inclusion of commands into blocks by a TTSM are expected to be tightly coupled to the block time of the chosen BCT. Methodologies from related work such as Prybila et al. [PSHW20], and Weber et al. [WXR+16] have been employed to further investigate latency imposed by the transaction inclusion duration[24] of the system. For the evaluation, it is assumed that all participants are available at any time.

As of section 4.2, commands in a TTSM are dispatched into the network of workflow participants and accepted or rejected by each of them. This is a two-step process which, if transactions are required to be included into a block of a blockchain before the TTSM can continue, requires at least two blocks to be produced — one in which the command is proposed and one in which participants accepted or rejected the command. The average transaction inclusion time is determined by the central limit theorem given in equation 5.8 where μ is the overall mean of the random samples $X_1, X_2, \ldots, X_n, \ldots$, while $\bar{X}_n$ is the mean of the first n samples and $\sigma_{\bar{X}} = \sigma/\sqrt{n}$ with σ^2 being the variance.

$$Z = \lim_{n \to \infty} \left(\frac{\bar{X}_n - \mu}{\sigma_{\bar{X}}} \right) \tag{5.8}$$

[22] Traces that can be used in a given workflow.
[23] Traces that cannot be used in a given workflow.
[24] Also referred to as consistency and persistence properties of a TTSM (see section 4.4).

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek WIEN Your knowledge hub

Given that completely finalizing a command takes at least two blocks, the median of the normal distribution is $\frac{3}{2} \cdot b_t$ where b_t is the average block time. This assumption is supported by the empirical evaluation results in table 5.3 equally for all types of commands across all scenarios. Regarding the simulation itself, at the time of writing this work, there are two proof-of-stake test networks available for Ethereum in particular, namely Sepolia[25] and Goerli[26]. In conjunction with Alchemy[27], a platform that provides tooling for web3 development, Goerli was used to run the simulations. All simulations used the same smart contract which was previously deployed — this is similar to how a TTSM would operate on the Ethereum main network in real-world scenarios as well. Afterwards, the workflow for each use case (facility management, supply chain and incident management) were created and instantiated exactly 50 times. After successful instantiation, each workflow was executed 10 times which results in the depicted number of samples in table 5.3 below.

	Facility Management		
	Samples	Av. Duration	Std. Dev. (σ)
Workflow Definition	50	18.209 s	3.8 s
Workflow Instantiation	50	17.847 s	3.6 s
State Transition	50	17.921 s	3.1 s
	Supply Chain		
	Samples	Av. Duration	Std. Dev. (σ)
Workflow Definition	50	17.383 s	3.5 s
Workflow Instantiation	50	18.619 s	3.9 s
State Transition	100	17.441 s	3.5 s
	Incident Management		
	Samples	Av. Duration	Std. Dev. (σ)
Workflow Definition	50	17.538 s	3.4 s
Workflow Instantiation	50	17.112 s	3.7 s
State Transition	90	17.985 s	3.3 s

Table 5.3: Average duration of each operation type

The results were extracted from multiple test runs of the TTSM prototype. Given that the block time of Ethereum, and in this case the Goerli test network, averages at around 12 seconds, it takes at least $\frac{3}{2} \cdot 12 = 18$ seconds to ensure that a command, including the participants responses, has been integrated into a new block on the blockchain. However, the standard deviation σ of the samples is rather large. This distribution originates from the randomization of execution times for each command resulting in a 12 to 24 seconds execution duration. Some commands have been dispatched immediately after a new block has been mined, resulting in an overall duration of around two block times (i.e., 24 seconds), while others have been dispatched at the end of a block and thus have been

[25]`https://sepolia.dev/` (accessed on 2022-11-25)
[26]`https://goerli.net/` (accessed on 2022-11-25)
[27]`https://alchemy.com/` (accessed on 2022-11-25)

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek WIEN Your knowledge hub

integrated into said block almost immediately after dispatching, resulting in a duration of just above one block time (i.e., > 12 seconds). The execution duration distribution is depicted as box plot in figure 5.11.

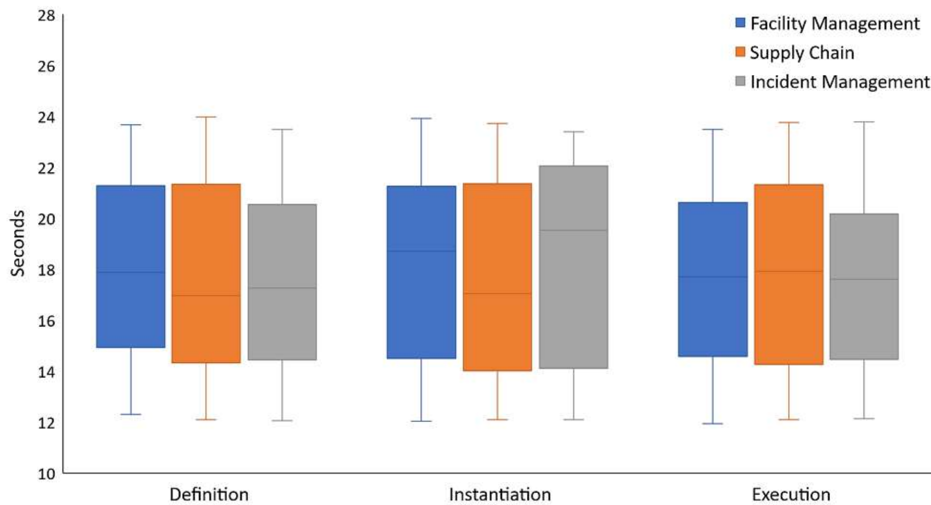

Figure 5.11: Box plot illustrating a one to two block time (12 s – 24 s) command duration imposed by the blockchain.

Nonetheless, the proposed TTSM concept technically also supports *optimistic workflow execution*. This kind of execution aims to eliminate the latency introduced by the blockchain. It enables participants to dispatch and distribute new commands to the workflow network without having to wait until the blockchain has produced a new block in which an integrity proof of the command is included. The aim of optimistic workflow execution[28] is to create a system that is close to the status quo of off-chain BPM engines regarding workflow execution duration and performance. While *enforced transaction inclusion*[29] halts workflow execution while waiting for the command integrity transaction to be included in the next block[30], *optimistic execution* immediately sends the command to all involved participants and afterwards waits for the blockchain response.

If the participant who dispatched the optimistic command eventually receives the transaction receipt, she creates a new consistency event that associates the optimistic command with the commitment reference from the receipt. Even though *optimistic execution* is much faster than *enforced inclusion*, participants must expect rollbacks at a later point in time when the blockchain catches up. If the integrity transaction cannot be included in a block on the blockchain or if any of the participants notice that an outstanding optimistic command is still missing its commitment reference after a predefined period of time, the command becomes invalid. This rolls the entire workflow state back to the

[28] Later on also referred to as *optimistic execution*.
[29] Later on also referred to as *enforced inclusion*.
[30] Which is required to generate a blockchain transaction receipt holding the commitment reference.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

point in time where it was still valid. The consistency and persistence events stored are still preserved inside the event bus for traceability reasons; however, the event sourcing systems of each particular module, which require a correct and consistent state, only accumulate events until the invalid one is reached. All events received after the rollback are then, once again, treated as valid ones.

Only a superficial *optimistic execution* implementation is provided in the TTSM prototype that eliminates the latency introduced by the blockchain but does not incorporate a rollback mechanism due to its complexity. Nonetheless, all scenarios have also been executed in this superficial *optimistic execution* mode. The evaluation results are given in table 5.4.

	Facility Management		
	Samples	Av. Duration	Std. Dev. (σ)
Workflow Definition	50	0.019 s	0.008 s
Workflow Instantiation	50	0.331 s	0.033 s
State Transition	50	0.114 s	0.055 s
	Supply Chain		
	Samples	Av. Duration	Std. Dev. (σ)
Workflow Definition	50	0.018 s	0.005 s
Workflow Instantiation	50	0.041 s	0.019 s
State Transition	100	0.079 s	0.033 s
	Incident Management		
	Samples	Av. Duration	Std. Dev. (σ)
Workflow Definition	50	0.019 s	0.005 s
Workflow Instantiation	50	0.023 s	0.006 s
State Transition	90	0.184 s	0.069 s

Table 5.4: Average duration of each operation type using *optimistic execution*

The execution duration per command has been reduced to the workflow logic and network latency itself, entirely omitting the delay imposed by the blockchain. Table 5.5 compares the total workflow execution duration of each scenario using *enforced transaction inclusion* and *optimistic workflow execution* modes.

Scenario	**Enforced Inclusion**			**Optimistic Execution**		
	Samples	Av. Dur.	σ	Samples	Av. Dur.	σ
Facility Management	10	57 s	3.1 s	10	0.57 s	0.3 s
Supply Chain	10	181 s	16.8 s	10	0.79 s	0.3 s
Incident Management	10	149 s	10.3 s	10	1.65 s	0.5 s

Table 5.5: Total duration of finality enforced and optimistic workflow execution

As illustrated before, choosing one execution mode over the other has a significant impact on the overall duration. When using *enforced inclusion* mode, waiting for the command

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

integrity transaction to be included in a new block on the Ethereum blockchain imposes a noteworthy latency. Figure 5.12 shows a pie chart depicting the duration required for the execution of the workflow logic (including network delays) and puts it against the downtime that transaction inclusion requires when explicitly enforced.

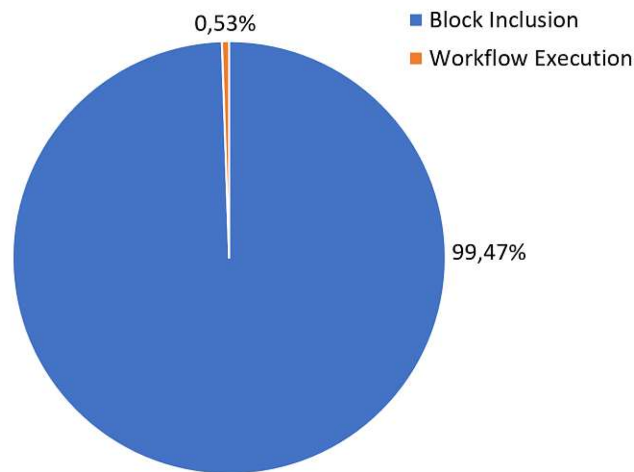

Figure 5.12: Pie chart illustrating a significant increase in workflow execution duration due to enforced transaction inclusion on the blockchain.

Relying on enforced transaction inclusion has a significant impact on practical usability. Therefore, it should primarily be used for moderately to slow-paced workflows (requiring no more than 2 to 3 state transitions per minute). On the other hand, optimistic execution should only be used for very fast-paced workflows (more than four state transitions per minute), where little to no conflicts are expected. Fast-paced workflows that do produce conflicts between participants (due to conflicting rules being employed on each side, for example) must expect frequent rollbacks, which halts execution for a brief moment. Another scenario applicable for optimistic execution are tasks that only involve one participant because no two parties can disagree on a new workflow state[31].

Nonetheless, it must be noted that these observations are only applicable when relying on an Ethereum-based consistency strategy. Optimistic execution might not even be considered viable when employing a strategy that, for example, leverages on a layer-2 rollup because the transaction inclusion duration is short enough to be neglected in the first place. Compared to existing approaches, a TTSM with an EVM-based strategy typically performs between 20% [WXR+16] and 25-times [PSHW20] better regarding execution duration.

[31] This also includes workflows only consisting of single party tasks (i.e., entirely private workflows).

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

Execution Cost

Unlike execution duration, execution cost is tightly coupled with the number of participants involved in a certain operation, as discussed in the static analysis in section 5.2.2. Even though the cost of creating a workflow definition, an instantiation of such, and performing state transitions is linear in growth, deploying the required smart contract that stores the hashes of these operations remains constant. The evaluation for execution cost was conducted according to the methodologies used by Weber et al. [WXR+16, GBPDW17], López-Pintado et al. [LPDGBW19], Sturm et al. [SSSJ19], Loukil et al. [LBAG21].

Table 5.6 shows the gas cost of each operation for each of the three scenarios. Given that the supply chain and incident management scenarios have the same number of participants but a different number of tasks (see table 5.1), one clearly identifies a strong correlation. The increasing costs come from the two-step process that each operation has to go through: (1) proposing a new definition, instantiation, or state transition, and (2) requiring all participants to send an accept or reject message.

	Facility Management
	Op. gas cost
Smart Contract	95,337
Workflow Definition	68,502
Workflow Instantiation	68,502
State Transition	68,502
	Supply Chain
	Op. gas cost
Smart Contract	95,337
Workflow Definition	137,004
Workflow Instantiation	137,004
State Transition	137,004
	Incident Management
	Op. gas cost
Smart Contract	95,337
Workflow Definition	137,004
Workflow Instantiation	137,004
State Transition	137,004

Table 5.6: Total gas cost of each operation type per scenario

Even though the cost for a single blockchain transaction is rather small, with 22,834 gas, scenarios involving a lot of participants may be at a disadvantage. Nonetheless, the TTSM prototype still performs better cost-wise than most other interpreted [LBAG21, LPDGBW19], and far better than most compiled approaches [WXR+16, GBPDW17] considering the three scenarios described above. Table 5.7 summarizes the total execution cost of each scenario, including smart contract deployment (even though only required

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Wien Bibliothek Your knowledge hub

once), workflow definition, instantiation, and all state transitions where each state transition is dispatched to every participant.

Scenario	Total gas cost
Facility Management	574,851
Supply Chain	1,739,385
Incident Management	1,602,381

Table 5.7: Total gas cost of each scenario

Notice that the total cost is still relatively high even though smart contract deployment is much more cost-efficient compared to existing approaches [LBAG21, LPDGBW19]. This gives an opportunity for future work to improve the concept of a TTSM by decoupling the execution cost from the number of involved participants. In particular, two areas for further research have been identified:

- Investigation of applicable techniques to reduce the number of participants involved in an operation to a minimum[32].
- Investigation of applicable blockchain-oriented software design patterns that allow decoupling of execution cost and the number of participants.

The first research challenge might require an in-depth analysis of statecharts and concepts such as the one proposed by Nakamura et al. [NMK18], where the interaction between participants is split into multiple statecharts. An interesting approach for the latter one, on the other hand, might be the off-chain signatures design pattern where all involved participants sign a proposed operation before writing the hash of the operation itself and the signatures to the blockchain [ET17, XWS19]. Others, such as Carminati et al. [CFR18], or Sun et al. [SYZ+21] motivate the usage of ZKPs, homomorphic encryption, or TEEs for concepts similar to a TTSM in order to improve privacy and confidentiality and, regarding the challenges mentioned above, to reduce execution cost. Nonetheless, the proposed concept and the TTSM prototype have shown their viability for use in real-world scenarios by demonstrating a rather low overall execution cost and exhibiting the potential for future improvements.

Practical Conflict Resolution

Scenarios such as the supply chain example described in section 5.3.2 are typically prone to create conflicts between participants. Resolving these, however, is rather trivial when properly employing a TTSM due to its consistency, persistence, and verifiability properties. To demonstrate this, a conflict was simulated for the supply chain scenario, where a deadline was not met. It was successfully resolved by leveraging the time-travelling

[32] Only the middleman and the carrier have to accept or reject a state transition if supplier, manufacturer, and buyer are not involved in this process step, for example.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

capabilities of the proposed concept and the prototype. By going back in time, the exact moment when delays were introduced was identified by the corresponding blockchain commitment reference. The hashes stored on-chain were then compared to the hashes of the operations stored off-chain. Therefore, the conflict was resolved by determining who introduced the delays by utilizing the auditability property of the blockchain.

Privacy and Flexibility

Privacy was demonstrated thoroughly throughout the simulation of all three scenarios. All data is stored in hashed form on the blockchain when using the EVM-strategy. Therefore, even on public blockchains, only participants involved in the workflow can read and verify workflow-specific data. Leveraging on the concept's flexibility, future work may replace the EVM-strategy with a strategy based on ZKPs, layer-2 rollups, or homomorphic encryption, for example. Strategies like these hold potential regarding sharing and proving specific properties fulfilled by private workflow tasks without exposing confidential information [CFR18, SYZ+21]. Especially rollups, where many try to establish EVM compatibility, would enable the portability of the proposed approach and the smart contract presented in code listing 5.1. Throughout the prototype implementation and evaluation, different strategies were employed to demonstrate the consistency module's flexibility. Nonetheless, metadata, such as hashes or ZKPs, still end up on the blockchain. Solving this issue is a topic for future research, especially regarding GDPR compliance.

5.4 Integration with Camunda's Zeebe

An aspect unneglectable in Design Science research is showing real-world utility of the produced artifacts [HMPR04]. One facet not discussed in this context until now is the integration of a TTSM into an existing and well-established BPM system to improve practical acceptance[33]. Therefore, this section briefly introduces a possible adaptation of the existing TTSM prototype in order to integrate it with Camunda's[34] workflow execution engine Zeebe[35], to then discuss further opportunities. Camunda cloud[36] was used to provide access to instances of Camunda and Zeebe.

5.4.1 Prototype Adaptations

An integration requires some minor adaptations in the form of extensions to the prototype. The *open-closed principle*, that the proposed concept for a TTSM strictly follows, together with the modular design and the established event bus system, make it straightforward to create and attach new subsystems [Mey97, Mar96, Gei15b]. A new module called *IntegrationsModule* has been added, to enable participants to dynamically integrate workflows

[33]Exchanging an existing solution with a new one is often times more complex and time consuming than integrating one into the other.

[34]Camunda Website (accessed on 2022-10-29)

[35]Camunda Docs: Zeebe (accessed on 2022-10-29)

[36]Camunda Cloud Console (accessed on 2022-10-29)

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

and workflow instances into existing BPM systems. It determines during runtime and, based on a given configuration which integrations to use. Furthermore, it also exposes the new *ZeebeModule*. The *ZeebeModule* is solely commissioned with the interaction between the TTSM prototype and Camunda's Zeebe workflow execution engine. It stores its local data in the form of event logs as part of the global event bus and accesses it by aggregation by using projections[37]. For the interaction between the prototype and Zeebe, the *ZeebeModule* relies on the **zeebe-node 8.1.2**[38] NodeJS library which allows the implementation of hooks that are triggered by Zeebe using gRPC. The code that links a TTSM workflow instance with a Zeebe process instance is shown in listing 5.3.

```
async linkProcessInstance(instance: WorkflowInstanceProposal) {

  // Job worker function to advance TTSM workflows.
  const handler: ZBWorkerTaskHandler = (job: ZeebeJob) => {
    try {
      this.workflowService.advanceWorkflowInstance(
        instance.id,
        {
          event: job.type,
          payload: job.variables
        }
      );
    } catch (error) {
      return job.fail(error.message);
    }
    return job.complete();
  };

  // Each workflow has a list of state names (strings).
  for (const nextStateName of instance.workflow.states) {

    // Register a job worker for each consistency task.
    this.zeebeClient.createWorker({
      taskType: nextStateName,
      taskHandler: handler
    });
  }
}
```

Listing 5.3: Implementation of dynamic Zeebe job worker registration

The *ZeebeModule* creates a new worker for each state in the given workflow. Whenever a participant advances the state inside Camunda, the corresponding worker is triggered and either returns a *fail* or *complete* response to Zeebe. Note that this is only a basic

[37]This is similar to the behavior of the rules module, for example.
[38]`https://npmjs.com/package/zeebe-node` (accessed on 2022-11-29)

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek WIEN Your knowledge hub

prototypical implementation of such an integration that should serve as proof of concept. It is not part of the actual TTSM concept itself. In a full-fledged implementation, the *ZeebeModule* should await inclusion of the transaction into a block on the blockchain before returning a response and notify all participants to keep the workflow state synchronized. The adaptations described above are also available on GitHub[39].

5.4.2 Workflow Execution

In order to leverage on the properties of a TTSM, the BPMN diagrams used in Camunda must follow a certain format. Each task in Camunda that wants to interact with the TTSM must be defined as a "service task" to trigger a job worker. These are tasks dedicated to ensure consistency and traceability using a TTSM and mark the interfaces between participants. An example of such a BPMN diagram is given in figure 5.13.

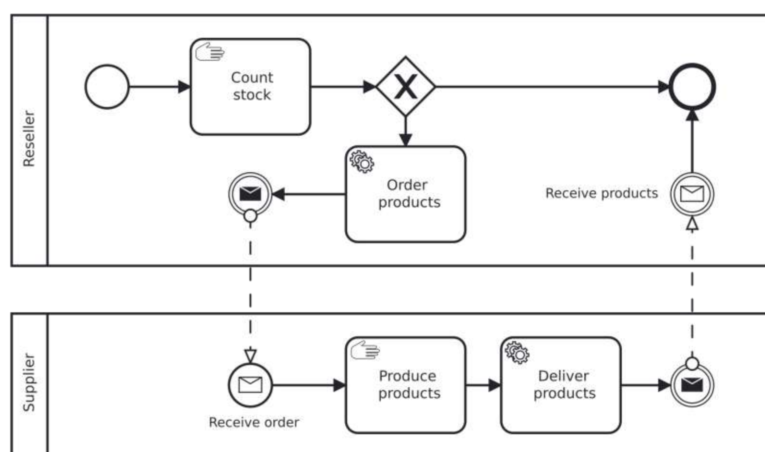

Figure 5.13: Exemplary reseller-supplier scenario with service tasks before each interaction.

A reseller counts her stock and, depending on the result, orders products from the supplier. The "Order products" activity is a service task that triggers a TTSM job worker, which handles the consistency concerns and the interaction between participants. After the supplier received the order and produced all required products, the deliver products service task is invoked, eventually leading to product delivery and completing the workflow. For the TTSM to properly interact with Camunda and Zeebe, a choreography diagram solely concerned with the interactions between participants must be derived. Such an integration-compliant diagram only contains the service tasks and the names of the participants. Other activities are treated as private. A dedicated consistency service task must be specified if private activities require consistency. This means that service tasks that trigger a job worker in a TTSM are not limited to interactions between two or more participants; however, they are predestined for it. Private service tasks require

[39]https://github.com/danielkleebinder/ttsm-prototype (accessed on 2022-11-29)

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

choreography activities to specify the same party as initiator and receiver. Figure 5.14 shows the choreography diagram of the reseller-supplier scenario from figure 5.13 above.

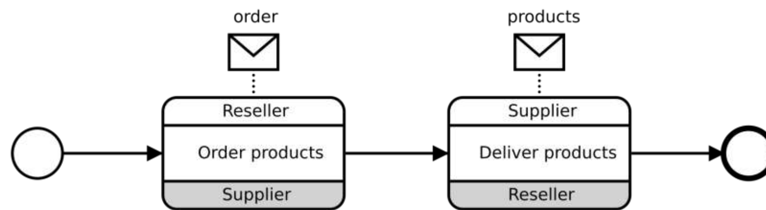

Figure 5.14: Choreography of the reseller-supplier scenario.

This choreography diagram is then converted to statecharts and deployed by the TTSM. Even though the deployment and instantiation are directly executed on the TTSM, the workflow instance execution is performed using the graphical UI of Camunda. All necessary configuration of the Zeebe-integration is part of the deployment and the instantiation of the workflow.

5.4.3 Summary and Discussion

The evaluation performed in this section has introduced some minor extensions to the TTSM prototype by adding a *ZeebeModule* for integration into Camunda's Zeebe workflow execution engine. It allows participants to execute workflow instances using the graphical UI of Camunda while keeping auditability, traceability, and consistency properties of a TTSM. However, the integration of workflow deployment was not possible. The evaluation of Camunda connectors[40] and Zeebe job workers[41] shows that they are no suitable solutions, because they are not concerned with the entirety of the workflow, but only with single tasks. This yields the following research challenge for future work:

- Investigation of integration strategies that allow participants to deploy and instantiate new workflows using Camunda and its underlying workflow execution engine Zeebe instead of relying on the TTSM.

Answering this research challenge might include writing a custom Camunda plugin[42]. Nonetheless, the integration seems to hold great potential. Therefore, a more in-depth investigation considering the integration of a TTSM into Camunda and Zeebe is advised as part of future work. This might include replacing parts of the Zeebe or Camunda architectures with concepts proposed in this work or vice versa by replacing the workflow module with Zeebe, for example. Even though the integration evaluation in this work has only been a proof of concept with very little functionality, it demonstrated interesting prospects that future work could leverage on.

[40] Camunda Docs: Connectors (accessed on 2022-10-29)
[41] Camunda Docs: Job Workers (accessed on 2022-10-29)
[42] Camunda Docs: Custom Plugins (accessed on 2022-10-29)

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek WIEN Your knowledge hub

5.4.4 Research Question 3

To answer the third research question:

> *Which aspects must be adapted to close the gap between the state of the art and a privacy-preserving BCT-based state machine that allows time-travel verification?*

Existing workflow execution systems already hold great potential when it comes to the integration of the concept for a TTSM. Not only is this a viable option when auditability or traceability are required, but the expenditure for such a venture also seems to be relatively low. Concepts proposed in related work that leverage on blockchains, however, have shown a formidable lack of flexibility and privacy. This can be traced back to the fact that almost all concepts rely on running code directly on layer-1 blockchains. Even though some concepts aim to circumnavigate or even solve the shortcomings imposed by layer-1 blockchains, neither can mitigate the majority of them. To close the gap between the state of the art and a privacy-preserving BCT–based state machine that allows time-travel verification, future concepts might want to start building on layer-2 or even start targeting layer-2 rollups and build on layer-3 instead of directly leveraging on blockchains. Using the blockchain as a supportive third-party system that provides certain properties increases the flexibility of the system built on top of it and reduces its cost. Nonetheless, the potential imposed by a BCT–based state machine that only takes advantage of blockchain properties without directly building, and thus constraining itself by it, is not to be neglected.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

CHAPTER 6

Conclusion

Computer-aided BPM is a hot topic for both industrial and research communities. With the uprise of blockchains in recent years, the latter started an endeavor towards blockchain-based BPM to leverage on one of its unique properties of BCTs — trust. Organizations and companies can now cooperate with each other without the need for a centralized, trusted third party. Nonetheless, research has been following a strict trend where software has been tightly coupled to the underlying blockchain resulting in a lack of flexibility and privacy. By performing a narrative literature review and applying design science, this work created a novel concept for building BPM systems that take advantage of the properties blockchains provide without restricting themselves by their shortcomings.

The concept proposes a four-module architecture where each module ensures specific properties. The *workflow module* is solely responsible for the creation and execution of workflows. By extending its interface, it can support a vast range of different BP modeling languages and is, therefore, very much declarative; a desirable property regarding the practical value of such systems. As the second one, the *rules module* allows participants to verify if rules beyond simple workflow semantics are fulfilled. The third module, the *consistency module*, provides an abstract interface to the underlying BCT. There are no strict requirements for this module. Usage of the Bitcoin network is as viable as the usage of layer-2 rollups, for example. The last module provides persistent storage and a convenient interface that allows participants to time travel between different workflow states. During the course of this work, the concept was continuously evaluated against qualitative metrics and predefined BPs to constantly improve its practicality.

The evaluation has shown significant traits regarding flexibility without forfeiting trust, traceability, or auditability provided by the underlying BCT. This was traced back to the TTSM operating one layer above the BCT without requiring tight coupling; a characteristic that has not yet been investigated in much detail in related literature. This allows for a highly modular architecture where components are only loosely coupled, interchangeable and extensible, resulting in a list of advantages, including the option to

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

switch between blockchains if needed, dynamic participant selection, or adjusting the workflow structure while being executed. Furthermore, during the execution of exemplary scenarios, it has been shown that, depending on the requirements of the participants, privacy can be fully preserved or only in parts if needed. In what detail conflicts between participants can be resolved then weakly correlates with the level of privacy.

Additionally, the proposed concept also reduces the overall execution cost of workflows. For the scenarios simulated in the evaluation, the TTSM prototype performed better than most existing solutions. However, the cost correlates linearly with the number of participants involved and will break even if this number exceeds a certain threshold. Nonetheless, there are promising BoSE software design patterns that can solve this issue. Further investigation in future work is advised.

Even though introducing an architecture that separates the blockchain and the workflow execution engine increases the system's complexity to some extent, it brings forth desirable properties, as mentioned above. Related literature and existing solutions, however, broadly introduce a tight coupling between both. Therefore, future work might want to investigate further into building blockchain-based BPM systems as layer-2 or layer-3 applications instead of directly leveraging on the blockchain and running code on layer-1. However, this requires preliminary work identifying desirable layer-1 properties and how they can be transferred and used on layer-2 or layer-3. In this context, one might extend and leverage upon the *consistency module* as proposed in this work.

Another topic future work should extend upon is the investigation of compensation mechanisms for lost peer-to-peer or even peer-to-blockchain connections. Solving this problem from the distributed systems domain further improves resiliency of the choreography but requires preliminary research of algorithms with small on-chain footprint that enables participants to verify the integrity of messages exchanged between participants.

Regarding the integration into other systems, a couple of open problems are still to answer. One is the handling of large or exotically structured payloads in state transitions. They cannot be part of the *persistence module* itself because it would clutter storage and transfer potentially unnecessary information between participants. A solution one might investigate is *content-addressable storage* where only the reference to the data is exchanged and stored on-chain. Another aspect to consider is the integration into existing BPM systems. Even though the integration into Camunda's Zeebe workflow execution engine has been shown to be viable during the evaluation, a thorough investigation is still outstanding.

Nonetheless, the idea of a TTSM that operates off-chain, but leverages upon the unique properties of BCTs, has been demonstrated. Especially improved flexibility is a trait future systems might build upon. This work should be considered as a starting point for BPM systems that take advantage of the blockchain as a source of trust, traceability, and auditability while treating it as a loosely coupled sub-system of supportive nature.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

List of Figures

2.1 A simplified business process of a maintenance contractor. 20
2.2 More complex two-lane diagram with interactions between participants. . . 21
2.3 Choreography diagram highlighting interactions between two participants. 22
2.4 Integration of a BPI into the two-party facility maintenance system. . . . 23
2.5 Integration of the Baseline Protocol for the exchange of the maintenance report between maintenance contractor and building administrator. 24

4.1 Macro architecture and event store design of a TTSM. 47
4.2 Life cycle of a command dispatched in a TTSM. 48
4.3 Sequence diagram for the creation of a workflow definition inside the workflow module. 50
4.4 Format of a consistency message exchanged by participants. 56
4.5 Setup of multiple TTSMs with counterparties interacting with each other. 56
4.6 UML class diagram of the workflow module of a prototypical TTSM. . . . 62
4.7 Persistence module and event store architecture. 66
4.8 UML class diagram of the rules module of a prototypical TTSM. 67
4.9 UML flowchart diagram of the rules modules decision-making process. . . 69
4.10 Consistency module architecture that enables multi-chain support. 70
4.11 Sequence diagram of two participants exchanging messages using the consistency module. 72

5.1 A classification for blockchain selection. 78
5.2 A classification for participant selection. 79
5.3 A classification for workflow mutability. 79
5.4 A classification for modelling language support. 80
5.5 A classification for sharing workflow structures. 81
5.6 A classification for sharing data between participants. 82
5.7 Mallory forwards messages from Alice to Bob. 87
5.8 BPMN diagram of the facility maintenance use case introduced in 2.4.1. . . 91
5.9 BPMN diagram of a supply chain adapted from Weber et al. [WXR+16]. 92
5.10 BPMN diagram of the incident management use case adapted from “BPMN 2.0 by Example” [OMG10]. 93
5.11 Box plot illustrating a one to two block time (12 s – 24 s) command duration imposed by the blockchain. 97

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

5.12 Pie chart illustrating a significant increase in workflow execution duration due to enforced transaction inclusion on the blockchain. 99
5.13 Exemplary reseller-supplier scenario with service tasks before each interaction. 104
5.14 Choreography of the reseller-supplier scenario. 105

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

List of Tables

3.1 Search terms used during the narrative review . . . 27
3.2 Comparison between different conceptual models and their properties . . 39

4.1 List of non-goals for the proposed concept . . . 46
4.2 List of workflow definition endpoints . . . 63
4.3 List of workflow instance endpoints . . . 63
4.4 List of rules endpoints . . . 68
4.5 List of required rule validation engine endpoints . . . 68

5.1 Structural comparison of the choreography diagrams and statecharts of the evaluated scenarios . . . 94
5.2 Average execution duration of each scenario . . . 95
5.3 Average duration of each operation type . . . 96
5.4 Average duration of each operation type using *optimistic execution* 98
5.5 Total duration of finality enforced and optimistic workflow execution . . . 98
5.6 Total gas cost of each operation type per scenario . . . 100
5.7 Total gas cost of each scenario . . . 101

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

List of Algorithms

4.1 Ensure optimization algorithm order 49
4.2 Rules checking algorithm 54
4.3 TTSM-consensus algorithm 59
4.4 Choosing a converter strategy 64

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

List of Listings

4.1 Exemplary workflow definition of a pedestrian traffic light 64
5.1 Implementation of a smart contract that stores message hashes 89
5.2 Implementation of the verification process of messages received 90
5.3 Implementation of dynamic Zeebe job worker registration 103

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

Acronyms

ABI Application Binary Interface. 14

BaaS Blockchain-as-a-Service. 31

BCT Blockchain Technology. 1–5, 27, 28, 40, 43–46, 51, 55, 60, 71, 73–78, 82–84, 95, 106–108

BIM Building Information Modelling. 51, 55, 67

BoSE Blockchain-oriented Software Engineering. 12, 108

BP Business Process. 19–25, 39–41, 43–45, 80, 81, 107

BPI Baseline Protocol Implementation. 23, 24, 109

BPM Business Process Management. 1–5, 19, 25, 27, 97, 102, 103, 107, 108

BPMN Business Process Model and Notation. 20–22, 28, 29, 33–36, 38–41, 45, 49, 58, 61, 63, 75, 90, 91, 93, 104

BPMS Business Process Management System. 29

CCSM Consensus Controlled State Machine. 23–25

CQRS Command-Query Responsibility Segregation. 52, 61, 74

DApp Distributed Application. 18

DLT Distributed Ledger Technology. 5, 9, 11, 12, 14

DTO Data Transfer Object. 62, 63

EDCC Executable Distributed Code Contract. 11, 12, 14, 15, 17, 46

EEA Enterprise Ethereum Alliance. 23

EVM Ethereum Virtual Machine. 89, 90, 99, 102

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

FSM Finite-State Machine. 49, 61

IoT Internet of Things. 1, 18

IPFS Interplanetary File System. 51

NFT Non-Fungible Token. 16

OOP Object-oriented Programming. 20

OoS out of scope. 46

PBFT Practical Byzantine Fault Tolerance. 10, 11

PoS Proof of Stake. 5, 7, 10, 11

PoW Proof of Work. 5–7, 10, 11

SLA Service-Level-Agreement. 29, 30

SMR State Machine Replication. 7

SOA Service-oriented Architecture. 31

TEE Trusted Execution Environment. 18, 101

TTSM Time-travelling State Machine. 43–53, 55–62, 64–77, 79–88, 91, 94–109, 113

UML Unified Modeling Language. 62, 67

UX User Experience. 14, 15

ZKP Zero-Knowledge Proof. 18, 24, 25, 56, 71, 81, 82, 84, 101, 102

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

Bibliography

[AGI+19] Marco Autili, Francesco Gallo, Paola Inverardi, Claudio Pompilio, and Massimo Tivoli. Introducing trust in service-oriented distributed systems through blockchain. In *2019 IEEE International Symposium on Software Reliability Engineering Workshops (ISSREW)*, pages 149–154, 2019.

[Agu10] Marcos K. Aguilera. *Stumbling over Consensus Research: Misunderstandings and Issues*, pages 59–72. Springer Berlin Heidelberg, Berlin, Heidelberg, 2010.

[BAA+15] Jim Barnett, Rahul Akolkar, RJ Auburn, Michael Bodell, Daniel C. Burnett, Jerry Carter, Scott McGlashan, Torbjörn Lager, Mark Helbing, Rafah Hosn, T.V. Raman, Klaus Reifenrath, No'am Rosenthal, and Johan Roxendal. State chart xml (scxml): State machine notation for control abstraction. [Online] Available: `https://www.w3.org/TR/scxml/`, 09 2015. (Accessed 2022-08-15).

[BDM+13] Dominic Betts, Julian Dominguez, Grigori Melnik, Fernando Simonazzi, and Mani Subramanian. *Exploring CQRS and Event Sourcing: A Journey into High Scalability, Availability, and Maintainability with Windows Azure*. Microsoft patterns & practices, 1st edition, 2013.

[Bel22] Adam Bellemare. Fact vs. delta event types. [Online] Available: `https://developer.confluent.io/learn-kafka/event-design/fact-vs-delta-events/`, 10 2022. (Accessed 2022-11-20).

[Ben14] Juan Benet. IPFS - content addressed, versioned, P2P file system. *CoRR*, abs/1407.3561, 2014.

[BF14] Pierre Bourque and Richard E. Fairley, editors. *SWEBOK: Guide to the Software Engineering Body of Knowledge*. IEEE Computer Society, Los Alamitos, CA, version 3.0 edition, 2014.

[BKM18] Ethan Buchman, Jae Kwon, and Zarko Milosevic. The latest gossip on BFT consensus. *CoRR*, abs/1807.04938, 2018.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek WIEN Your knowledge hub

[Bra12] Jamie Brandon. Causal ordering. [Online] Available: `https://www.scattered-thoughts.net/writing/causal-ordering/`, 08 2012. (Accessed 2022-08-21).

[Bro19] Paul Brody. How public blockchains are making private blockchains obsolete. [Online] Available: `https://go.ey.com/2EbwphF`, 12 2019. (Accessed 2022-10-17).

[BSA14] Alysson Bessani, João Sousa, and Eduardo E.P. Alchieri. State machine replication for the masses with bft-smart. In *2014 44th Annual IEEE/IFIP International Conference on Dependable Systems and Networks*, pages 355–362, 2014.

[BSH+20] Florian Blum, Benedikt Severin, Michael Hettmer, Philipp Hückinghaus, and Volker Gruhn. Building hybrid dapps using blockchain tactics -the meta-transaction example. In *2020 IEEE International Conference on Blockchain and Cryptocurrency (ICBC)*, pages 1–5, 2020.

[But22] Vitalik Buterin. A next-generation smart contract and decentralized application platform. [Online] Available: `https://ethereum.org/en/whitepaper/`, 10 2022. (Accessed 2022-10-17).

[Cam22] Cambridge Bitcoin electricity consumption index. [Online] Available: `https://cbeci.org/`, 10 2022. (Accessed 2022-10-05).

[Car22] J.D. Carlston. What's in an (event) name? [Online] Available: `https://www.eventstore.com/blog/whats-in-an-event-name`, 01 2022. (Accessed 2022-10-17).

[CFR18] Barbara Carminati, Elena Ferrari, and Christian Rondanini. Blockchain as a platform for secure inter-organizational business processes. In *2018 IEEE 4th International Conference on Collaboration and Internet Computing (CIC)*, pages 122–129, 2018.

[CK92] Michael Christel and Kyo Kang. Issues in requirements elicitation. Technical Report CMU/SEI-92-TR-012, Software Engineering Institute, Carnegie Mellon University, Pittsburgh, PA, 09 1992.

[CL99] Miguel Castro and Barbara Liskov. Practical byzantine fault tolerance. *OSDI*, 03 1999.

[CRF18] Barbara Carminati, Christian Rondanini, and Elena Ferrari. Confidential business process execution on blockchain. In *2018 IEEE International Conference on Web Services (ICWS)*, pages 58–65, 2018.

[dAS19] Monika di Angelo and Gernot Salzer. A survey of tools for analyzing ethereum smart contracts. In *2019 IEEE International Conference on*

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

Decentralized Applications and Infrastructures (DAPPCON), pages 69–78, 2019.

[Den97] Peter J. Denning. A new social contract for research. *Communications of the ACM*, 40:132–134, 02 1997.

[EEA22] Baseline Protocol Specifications. [Online] Available: `https://github.com/eea-oasis/baseline-standard`, 09 2022. (Accessed 2022-10-17).

[EH18] Jacob Eberhardt and Jonathan Heiss. Off-chaining models and approaches to off-chain computations. In *Proceedings of the 2nd Workshop on Scalable and Resilient Infrastructures for Distributed Ledgers*, 12 2018.

[ES13] Ittay Eyal and Emin Gün Sirer. Majority is not enough: Bitcoin mining is vulnerable. *CoRR*, abs/1311.0243, 2013.

[ET17] Jacob Eberhardt and Stefan Tai. *On or Off the Blockchain? Insights on Off-Chaining Computation and Data*, pages 3–15. Service-Oriented and Cloud Computing. Springer International Publishing, 2017.

[FLP85] Michael J. Fischer, Nancy A. Lynch, and Michael S. Paterson. Impossibility of distributed consensus with one faulty process. *J. ACM*, 32(2):374—-382, April 1985.

[FRMKR15] Walid Fdhila, Stefanie Rinderle-Ma, David Knuplesch, and Manfred Reichert. Change and compliance in collaborative processes. In *2015 IEEE International Conference on Services Computing*, pages 162–169, 2015.

[FRMR12] Walid Fdhila, Stefanie Rinderle-Ma, and Manfred Reichert. Change propagation in collaborative processes scenarios. In *8th International Conference on Collaborative Computing: Networking, Applications and Worksharing (CollaborateCom)*, pages 452–461, 01 2012.

[FSM21] Abdmeziem Farah, Boukhedouma Saida, and Oussalah Chabane Mourad. On the security of business processes: classification of approaches, comparison, and research directions. In *2021 International Conference on Networking and Advanced Systems (ICNAS)*, pages 1–8, 2021.

[FVB05] Brian Fynes, Chris Voss, and Seán Búrca. The impact of supply chain relationship quality on quality performance. *International Journal of Production Economics*, 96:339–354, 02 2005.

[FWSZ21] Xiang Fu, Huaimin Wang, Peichang Shi, and Xunhui Zhang. Teegraph: A blockchain consensus algorithm based on tee and dag for data sharing in iot. *Journal of Systems Architecture*, 122:102344, 11 2021.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

[GBPDW17] Luciano García-Bañuelos, Alexander Ponomarev, Marlon Dumas, and Ingo Weber. Optimized execution of business processes on blockchain. In Josep Carmona, Gregor Engels, and Akhil Kumar, editors, *Business Process Management*, pages 130–146, Cham, 2017. Springer International Publishing.

[Gei15a] Matthias Geirhos. *Entwurfsmuster – Das umfassende Handbuch*. Rheinwerk Computing, Rheinwerkallee 4, 53227 Bonn, Germany, 2015.

[Gei15b] Matthias Geirhos. *Entwurfsmuster – Das umfassende Handbuch*. Rheinwerk Computing, Rheinwerkallee 4, 53227 Bonn, Germany, 2015.

[Gei15c] Matthias Geirhos. Verhaltensmuster. In *Entwurfsmuster – Das umfassende Handbuch* [Gei15b], pages 343–350.

[GGSGL+20] Julian Alberto Garcia-Garcia, Nicolás Sánchez-Gómez, David Lizcano, M. J. Escalona, and Tomás Wojdyński. Using blockchain to improve collaborative business process management: Systematic literature review. *IEEE Access*, 8:142312–142336, 2020.

[Har87] David Harel. Statecharts: A visual formalism for complex systems. *Sci. Comput. Program.*, 8(3):231—274, 06 1987.

[HMPR04] Alan R. Hevner, Salvatore T. March, Jinsoo Park, and Sudha Ram. Design science in information systems research. *MIS Quarterly*, 28:75–105, 03 2004.

[ISO04] Data elements and interchange formats – information interchange – representation of dates and times. [Online] Available: `https://www.iso.org/standard/40874.html`, 2004. (Accessed 2022-09-05).

[JBS+20] Chandra Priya Jayabal, Ponsy Bhama, S. Swarnalaxmi, A. Safa, and I. Elakkiya. *Blockchain Centered Homomorphic Encryption: A Secure Solution for E-Balloting*, pages 811–819. Springer International Publishing, 01 2020.

[JMSK04] David A Johnston, David M McCutcheon, F.Ian Stuart, and Hazel Kerwood. Effects of supplier trust on performance of cooperative supplier relationships. *Journal of Operations Management*, 22(1):23–38, 2004.

[KJ21] Ilyass El Kassmi and Zahi Jarir. Blockchain-oriented inter-organizational collaboration between healthcare providers to handle the covid-19 process. *International Journal of Advanced Computer Science and Applications*, 12:762–780, 2021.

[Kle17] Martin Kleppmann. *Designing data-intensive applications: The big ideas behind reliable, scalable, and maintainable systems*. "O'Reilly Media, Inc.", 2017.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

[Lam78] Leslie Lamport. Time, clocks, and the ordering of events in a distributed system. *Commun. ACM*, 21(7):558—-565, July 1978.

[LBAG21] Faiza Loukil, Khouloud Boukadi, Mourad Abed, and Chirine Ghedira. Decentralized collaborative business process execution using blockchain. *World Wide Web*, page 19, 09 2021.

[LFW20] Jan Ladleif, Christian Friedow, and Mathias Weske. An architecture for multi-chain business process choreographies. In Witold Abramowicz and Gary Klein, editors, *Business Information Systems*, pages 184–196, Cham, 2020. Springer International Publishing.

[LPDGBW19] Orlenys López-Pintado, Marlon Dumas, Luciano García-Bañuelos, and Ingo Weber. Interpreted execution of business process models on blockchain. In *2019 IEEE 23rd International Enterprise Distributed Object Computing Conference (EDOC)*, pages 206–215, 2019.

[LPGBD+19] Orlenys López-Pintado, Luciano García-Bañuelos, Marlon Dumas, Ingo Weber, and Alexander Ponomarev. Caterpillar: A business process execution engine on the ethereum blockchain. *Software: Practice and Experience*, 49:1162–1193, 2019.

[LPGBDW17] Orlenys López-Pintado, Luciano García-Bañuelos, Marlon Dumas, and Ingo Weber. Caterpillar: A blockchain-based business process management system. In *BPM*, 2017.

[LSNW20] Tom Lichtenstein, Simon Siegert, Adriatik Nikaj, and Mathias Weske. Data-driven process choreography execution on the blockchain: A focus on blockchain data reusability. In Witold Abramowicz and Gary Klein, editors, *Business Information Systems*, pages 224–235, Cham, 2020. Springer International Publishing.

[LSP02] Leslie Lamport, Robert Shostak, and Marshall Pease. The byzantine generals problem. *ACM Trans. Program. Lang. Syst.*, 4, 02 2002.

[LWW19] Jan Ladleif, Mathias Weske, and Ingo Weber. Modeling and enforcing blockchain-based choreographies. In Thomas Hildebrandt, Boudewijn F. van Dongen, Maximilian Röglinger, and Jan Mendling, editors, *Business Process Management*, pages 69–85, Cham, 2019. Springer International Publishing.

[Mar96] Robert Cecil Martin. The open-closed principle. [Online] Available: `https://courses.cs.duke.edu/fall07/cps108/papers/ocp.pdf`, 01 1996. (Accessed 2022-10-29).

[Mar17] Robert C. Martin. *Clean Architecture: A Craftsman's Guide to Software Structure and Design.* Prentice Hall Press, USA, 1st edition, 2017.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

[Mey88] Bertrand Meyer. Eiffel: A language and environment for software engineering. *Journal of Systems and Software*, 8(3):199–246, 1988.

[Mey97] Bertrand Meyer. *Object-Oriented Software Construction (2nd Ed.)*. Prentice-Hall, Inc., USA, 1997.

[MWA+18] Jan Mendling, Ingo Weber, Wil Van Der Aalst, Jan Vom Brocke, Cristina Cabanillas, Florian Daniel, Søren Debois, Claudio Di Ciccio, Marlon Dumas, Schahram Dustdar, Avigdor Gal, Luciano García-Bañuelos, Guido Governatori, Richard Hull, Marcello La Rosa, Henrik Leopold, Frank Leymann, Jan Recker, Manfred Reichert, Hajo A. Reijers, Stefanie Rinderle-Ma, Andreas Solti, Michael Rosemann, Stefan Schulte, Munindar P. Singh, Tijs Slaats, Mark Staples, Barbara Weber, Matthias Weidlich, Mathias Weske, Xiwei Xu, and Liming Zhu. Blockchains for business process management - challenges and opportunities. *ACM Trans. Manage. Inf. Syst.*, 9(1), 02 2018.

[Nak09] Satoshi Nakamoto. Bitcoin: A peer-to-peer electronic cash system. *Cryptography Mailing list at https://metzdowd.com*, 03 2009. (Accessed 2021-09-15).

[New19] Sam Newman. Just enough microservices. In *Monolith to Microservices: Evolutionary Patterns to Transform Your Monolith*, pages 1–32. O'Reilly Media, Incorporated, 1005 Gravenstein Highway North, Sebastopol, CA 95472, USA, 11 2019.

[NMK18] Hiroaki Nakamura, Kohtaroh Miyamoto, and Michiharu Kudo. Inter-organizational business processes managed by blockchain. In Hakim Hacid, Wojciech Cellary, Hua Wang, Hye-Young Paik, and Rui Zhou, editors, *Web Information Systems Engineering – WISE 2018*, pages 3–17, Cham, 2018. Springer International Publishing.

[OMG10] Bpmn 2.0 by example. [Online] Available: `http://docenti.ing.unipi.it/m.cimino/gpa/res/BPMN_by_example.pdf`, 06 2010. (Accessed 2022-10-08).

[OMG11] Business Process Model and Notation (BPMN) - Version 2.0. [Online] Available: `https://www.omg.org/spec/BPMN/2.0/PDF`, 01 2011. (Accessed 2022-04-20).

[OMT09] Advanced Trusted Environment: OMTP TR1. [Online] Available: `http://www.omtp.org/OMTP_Advanced_Trusted_Environment_OMTP_TR1_v1_1.pdf`, 05 2009. (Accessed 2022-10-31).

[Pel03] C. Peltz. Web services orchestration and choreography. *Computer*, 36(10):46–52, 2003.

[PG17] Ross C. Phillips and Denise Gorse. Predicting cryptocurrency price bubbles using social media data and epidemic modelling. In *2017 IEEE Symposium Series on Computational Intelligence (SSCI)*, pages 1–7, 11 2017.

[PL09] Photis Panayides and Y.H. Lun. The impact of trust on innovativeness and supply chain performance. *International Journal of Production Economics*, 122:35–46, 11 2009.

[PPMT17] Simone Porru, Andrea Pinna, Michele Marchesi, and Roberto Tonelli. Blockchain-oriented software engineering: Challenges and new directions. In *2017 IEEE/ACM 39th International Conference on Software Engineering Companion (ICSE-C)*, pages 169–171, 2017.

[Pry16] Christoph Prybila. Runtime verification for business processes utilizing the blockchain. Master's thesis, TU Wien, 1040 Wien, Austria, 12 2016.

[Pry19] Nat Pryce. Mistakes we made adopting event sourcing (and how we recovered). [Online] Available: `http://natpryce.com/articles/000819.html`, 06 2019. (Accessed 2022-09-06).

[PSHW20] Christoph Prybila, Stefan Schulte, Christoph Hochreiner, and Ingo Weber. Runtime verification for business processes utilizing the bitcoin blockchain. *Future Generation Computer Systems*, 107:816–831, 2020.

[RA11] Ellen Rhoades and Ellen A. Literature reviews. *The Volta Review*, 111:354–369, 09 2011.

[RCDF20] Christian Rondanini, Barbara Carminati, Federico Daidone, and Elena Ferrari. Blockchain-based controlled information sharing in inter-organizational workflows. In *2020 IEEE International Conference on Services Computing (SCC)*, pages 378–385, 11 2020.

[RJ20] C. Rosenthal and N. Jones. *Chaos Engineering: System Resiliency in Practice*. O'Reilly Media, 2020.

[Sal20] Fahad Saleh. Blockchain without waste: Proof-of-stake. *The Review of Financial Studies*, 34, 07 2020.

[SAW20] Siti Saadah and A.A Ahmad Whafa. Monitoring financial stability based on prediction of cryptocurrencies price using intelligent algorithm. In *2020 International Conference on Data Science and Its Applications (ICoDSA)*, pages 1–10, 08 2020.

[SB12] João Sousa and Alysson Bessani. From byzantine consensus to bft state machine replication: A latency-optimal transformation. In *2012 Ninth European Dependable Computing Conference*, pages 37–48, 2012.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

[SBK+17] Marc Stevens, Elie Bursztein, Pierre Karpman, Ange Albertini, and Yarik Markov. The first collision for full sha-1. In Jonathan Katz and Hovav Shacham, editors, *Advances in Cryptology – CRYPTO 2017*, pages 570–596, Cham, 2017. Springer International Publishing.

[SLHK19] A. Shahaab, B. Lidgey, C. Hewage, and I. Khan. Applicability and appropriateness of distributed ledgers consensus protocols in public and private sectors: A systematic review. *IEEE Access*, 7:43622–43636, 2019.

[Som10] Ian Sommerville. *Software Engineering*. Addison-Wesley, Harlow, England, 9 edition, 2010.

[SSSJ19] Christian Sturm, Jonas Szalanczi, Stefan Schönig, and Stefan Jablonski. *A Lean Architecture for Blockchain Based Decentralized Process Execution*, pages 361–373. Springer International Publishing, Cham, 01 2019.

[Str19] Samuel Stratton. Literature reviews: Methods and applications. *Prehospital and Disaster Medicine*, 34:347–349, 08 2019.

[SYZ+21] Xiaoqiang Sun, F. Richard Yu, Peng Zhang, Zhiwei Sun, Weixin Xie, and Xiang Peng. A survey on zero-knowledge proof in blockchain. *IEEE Network*, 35(4):198–205, 2021.

[Vog09] Werner Vogels. Eventually consistent. *Commun. ACM*, 52(1):40—-44, 01 2009.

[VXBP19] Wattana Viriyasitavat, Li Da Xu, Zhuming Bi, and Vitara Pungpapong. Blockchain and internet of things for modern business process in digital economy—the state of the art. *IEEE Transactions on Computational Social Systems*, 6(6):1420–1432, 2019.

[WEHG18] Florian Wessling, Christopher Ehmke, Marc Hesenius, and Volker Gruhn. How much blockchain do you need? towards a concept for building hybrid dapp architectures. In *2018 IEEE/ACM 1st International Workshop on Emerging Trends in Software Engineering for Blockchain (WETSEB)*, pages 44–47, 2018.

[WEMG19] Florian Wessling, Christopher Ehmke, Ole Meyer, and Volker Gruhn. Towards blockchain tactics: Building hybrid decentralized software architectures. In *2019 IEEE International Conference on Software Architecture Companion (ICSA-C)*, pages 234–237, 2019.

[Wes12a] Mathias Weske. *Business Process Management Architectures*, pages 333–371. Springer Berlin Heidelberg, Berlin, Heidelberg, 2012.

[Wes12b] Mathias Weske. *Introduction*, pages 3–23. Springer Berlin Heidelberg, Berlin, Heidelberg, 2012.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

[Wes12c] Mathias Weske. *Process Choreographies*, pages 243–291. Springer Berlin Heidelberg, Berlin, Heidelberg, 2012.

[Wes12d] Mathias Weske. *Process Orchestrations*, pages 125–242. Springer Berlin Heidelberg, Berlin, Heidelberg, 2012.

[Wes12e] Mathias Weske. *Properties of Business Processes*, pages 293–329. Springer Berlin Heidelberg, Berlin, Heidelberg, 2012.

[WG18] Florian Wessling and Volker Gruhn. Engineering software architectures of blockchain-oriented applications. In *2018 IEEE International Conference on Software Architecture Companion (ICSA-C)*, pages 45–46, 2018.

[Woo22] Gavin Wood. Ethereum: A secure decentralised generalised transaction ledger. [Online] Available: `https://github.com/ethereum/yellowpaper`, 08 2022. (Accessed 2022-10-17).

[WXR+16] Ingo Weber, Xiwei Xu, Régis Riveret, Guido Governatori, Alexander Ponomarev, and Jan Mendling. Untrusted business process monitoring and execution using blockchain. In Marcello La Rosa, Peter Loos, and Oscar Pastor, editors, *Business Process Management*, pages 329–347, Cham, 2016. Springer International Publishing.

[WZ18] Maximilian Wohrer and Uwe Zdun. Smart contracts: security patterns in the ethereum ecosystem and solidity. In *2018 International Workshop on Blockchain Oriented Software Engineering (IWBOSE)*, pages 2–8, 2018.

[XWS+17] Xiwei Xu, Ingo Weber, Mark Staples, Liming Zhu, Jan Bosch, Len Bass, Cesare Pautasso, and Paul Rimba. A taxonomy of blockchain-based systems for architecture design. In *2017 IEEE International Conference on Software Architecture (ICSA)*, pages 243–252, 2017.

[XWS19] Xiwei Xu, Ingo Weber, and Mark Staples. *Blockchain Patterns*, pages 113–148. Springer International Publishing, Cham, 2019.

[YJD09] Wang Yu, Chen Jianhua, and He Debiao. A new collision attack on md5. In *2009 International Conference on Networks Security, Wireless Communications and Trusted Computing*, volume 2, pages 767–770, 2009.

[ZHJ04] Tewfik Ziadi, Loïc Hélouët, and Jean-Marc Jézéquel. Revisiting statechart synthesis with an algebraic approach. In *Proceedings. 26th International Conference on Software Engineering*, volume 26, pages 242–251, 06 2004.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub